NASA-CR-192636

DEVELOPMENT OF ITERATIVE TECHNIQUES FOR THE SOLUTION OF UNSTEADY COMPRESSIBLE VISCOUS FLOWS

Grant NAG-1-1217

P.162

Final Report covering the Period
February 14, 1991 - February 13, 1993

Submitted to

NASA Langley Research Center
Hampton, VA 23665

Attn: Dr. Woodrow Whitlow

Prepared by

Duane Hixon, Graduate Research Assistant
L. N. Sankar, Professor
School of Aerospace Engineering
Georgia Institute of Tevhnology, Atlanta, GA 30332-0150

March 1993

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

During the past two decades, there has been significant progress in the field of numerical simulation of unsteady compressible viscous flows. At present, a variety of solution techniques exist such as the transonic small disturbance analyses (TSD) [e.g. Ref. 1-3], transonic full potential equation-based methods [e.g. Ref. 4-6], unsteady Euler solvers [e.g. Ref. 7-8], and unsteady Navier-Stokes solvers [e.g. Ref. 9-12]. These advances have been made possible by developments in three areas: (1) Improved numerical algorithms, (2) Automation of body-fitted grid generation schemes, and (3) Advanced computer architectures with vector processing and massively parallel processing features.

Despite these advances, numerical simulation of unsteady viscous flows still remains a computationally intensive problem, even in two dimensions. For example, the problem of dynamic stall of an oscillating NACA 0012 airfoil using state of the art alternating direction implicit (ADI) procedures presently require between 10,000 and 20,000 time steps per cycle of oscillation at low reduced frequencies when the viscous flow region is sufficiently resolved [Ref. 9]. In three dimensions, unsteady Navier-Stokes simulations of a helicopter rotor blade in forward flight requires over 30,000 time steps or more for a full revolution of the rotor [Ref. 10]. In other unsteady flows, such as the high angle of attack flow past fighter aircraft configurations, a systematic parametric study of the flow is presently not practical due to the very large CPU time needed for the simulations [Ref. 13]. Thus, it is clear that significant improvements to the existing algorithms, or

dramatic improvements in computer architectures will be needed, before unsteady viscous flow analyses become practical day-to-day engineering tools.

One scheme that has been of recent interest is the Generalized Minimal RESidual (GMRES) method originally proposed by Saad and Schultz (Ref. 14). This procedure uses a conjugate gradient method to accelerate the convergence of existing flow solvers. GMRES was added to existing steady flow solvers by Wigton, Yu, and Young (Ref. 15), and to an unstructured grid flow solver by Venkatakrishnan and Mavriplis (Ref. 16). Saad has also used a similar Krylov subspace projection method on a steady, incompressible Navier-Stokes problem and an unsteady one dimensional wave propagation equation (Ref. 17). To our knowledge, GMRES has not been applied to multi-dimensional unsteady compressible flow problems.

In this work, the GMRES scheme has been considered as a candidate for acceleration of a Newton iteration time marching scheme for unsteady 2-D and 3-D compressible viscous flow calculation; from preliminary calculations, this will provide up to a 65% reduction in the computer time requirements over the existing class of explicit and implicit time marching schemes. The proposed method has been tested on structured grids, but is flexible enough for extension to unstructured grids. The described scheme has been tested only on the current generation of vector processor architectures of the Cray Y/MP class, but should be suitable for adaptation to massively parallel machines.

CHAPTER II

MATHEMATICAL FORMULATION OF THE NAVIER-STOKES EQUATIONS

This work is mainly concerned with calculating unsteady, viscous, compressible flow. Thus, the full Navier-Stokes equations are solved in two or three dimensions.

In the following sections, the two dimensional Navier-Stokes equations are developed in both Cartesian and curvilinear coordinates, and then extended to 3-D.

2-D Governing Equations in Cartesian Coordinates

Fluid motion can be described using the concept that mass, momentum, and energy are conserved. Given this, the governing equations of fluid flows can be written in a variety of different forms. The form used in this work is the conservation form, which is written in general form as:

$$\langle q \rangle_t + \langle E \rangle_x + \langle G \rangle_z = 0 \qquad (2.1)$$

The conservation form is used for this work because numerical methods based on the nonconservation form may not conserve flux properties properly across physical discontinuities such as shocks. The nondimensionalized Navier-Stokes equations may be written in conservation form as:

$$\partial_t q + \partial_x E + \partial_z G = \frac{1}{Re}\left\langle \partial_x R + \partial_z T \right\rangle \tag{2.2}$$

where

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho w \\ e \end{pmatrix} \quad E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u w \\ u\langle e+p \rangle \end{pmatrix} \quad G = \begin{pmatrix} \rho w \\ \rho u w \\ \rho w^2 + p \\ w\langle e+p \rangle \end{pmatrix} \quad R = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xz} \\ R_4 \end{pmatrix} \quad T = \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{zz} \\ T_4 \end{pmatrix} \tag{2.3}$$

and:

$$\begin{aligned} \tau_{xx} &= \langle \lambda + 2\mu \rangle u_x + \lambda w_z \\ \tau_{xz} &= \mu \langle u_z + w_x \rangle \\ \tau_{zz} &= \langle \lambda + 2\mu \rangle w_z + \lambda u_x \\ R_4 &= u\tau_{xx} + w\tau_{xz} + \frac{\mu}{Pr\langle \gamma - 1 \rangle}\partial_x a^2 \\ T_4 &= u\tau_{xz} + w\tau_{zz} + \frac{\mu}{Pr\langle \gamma - 1 \rangle}\partial_z a^2 \end{aligned} \tag{2.4}$$

Also,

$$p = \langle \gamma - 1 \rangle \left[e - \frac{1}{2}\rho \langle u^2 + w^2 \rangle \right]$$

$$a^2 = \frac{\gamma p}{\rho} = \gamma(\gamma-1)\left[\frac{e}{\rho} - \frac{1}{2}(u^2 + w^2)\right] \qquad (2.5)$$

These are the 2-D Navier-Stokes equations for viscous, compressible fluid flow. In these equations, ρ is the density, u and w are the velocity components, e is the total energy per unit volume, a is the speed of sound for a perfect gas, p is the pressure, and γ is the ratio of specific heats (which is taken to be 1.4). The Reynolds number is Re and the Prandtl number is Pr. Using Stokes' Hypothesis, the bulk viscosity coefficient λ is given as:

$$\lambda = -\frac{2}{3}\mu \qquad (2.6)$$

For high Reynolds number flow, where turbulence occurs, the dependent variables are decomposed into steady and fluctuating components, and the equations are time averaged. The time averaging process gives rise to new terms which may be interpreted as turbulent stresses acting on the fluid. These are called Reynolds stresses. Using Boussinesq's concept, these stresses can be related to the rate of mean strain by means of an eddy viscosity. With this, an effective viscosity, consisting of the laminar viscosity and a computed eddy viscosity, is defined. More information on the eddy viscosity formulation is given in the section on numerical formulation.

To obtain the inviscid (Euler) flow equations, the right hand side of Eq. (2.2), which contains the viscous terms, is removed.

The Navier-Stokes equations have been nondimensionalized by scaling the dependent variables by the freestream density and speed of sound and the independent variables by the chord length c:

$$\tilde{t} = \frac{t}{\left(\frac{a_\infty}{c}\right)}$$

$$\tilde{x} = \frac{x}{c}$$

$$\tilde{z} = \frac{z}{c}$$

$$\tilde{\rho} = \frac{\rho}{\rho_\infty}$$

$$\tilde{u} = \frac{u}{a_\infty}$$

$$\tilde{w} = \frac{w}{a_\infty}$$

$$\tilde{e} = \frac{e}{\left\langle \rho_\infty a_\infty^2 \right\rangle} \qquad (2.7)$$

With this choice of nondimensionalization parameters, the Reynolds number is given as:

$$Re = \frac{\rho_\infty a_\infty c}{\mu_\infty} \qquad (2.8)$$

Note that this Reynolds number is based on the freestream speed of sound, and thus it must be scaled by the freestream Mach number to match a Reynolds number based on the freestream velocity.

From this point, all quantities will be nondimensionalized, so the tilde (~) will be dropped.

3-D Governing Equations in Cartesian Coordinates

The nondimensionalized 3-D Navier-Stokes equations are written in conservation form as:

$$\partial_t q + \partial_x E + \partial_y F + \partial_z G = \frac{1}{Re}\left\langle \partial_x R + \partial_y S + \partial_z T \right\rangle \tag{2.9}$$

where:

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{pmatrix} \quad E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ u\langle e + p \rangle \end{pmatrix} \quad F = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ v\langle e + p \rangle \end{pmatrix} \quad G = \begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ w\langle e + p \rangle \end{pmatrix}$$

$$R = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ R_5 \end{pmatrix} \quad S = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ S_5 \end{pmatrix} \quad T = \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ T_5 \end{pmatrix} \tag{2.10}$$

and

$$\tau_{xx} = \langle \lambda + 2\mu \rangle u_x + \lambda v_y + \lambda w_z$$
$$\tau_{xy} = \mu \langle u_y + v_x \rangle$$
$$\tau_{xz} = \mu \langle u_z + w_x \rangle$$
$$\tau_{yy} = \lambda u_x + \langle \lambda + 2\mu \rangle v_y + \lambda w_z$$
$$\tau_{yz} = \mu \langle v_z + w_y \rangle$$
$$\tau_{zz} = \lambda u_x + \lambda v_y + \langle \lambda + 2\mu \rangle w_z$$
$$R_5 = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \frac{\mu}{Pr\langle \gamma - 1 \rangle} \partial_x a^2$$
$$S_5 = u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + \frac{\mu}{Pr\langle \gamma - 1 \rangle} \partial_y a^2$$
$$T_5 = u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + \frac{\mu}{Pr\langle \gamma - 1 \rangle} \partial_z a^2 \qquad (2.11)$$

In 3-D,

$$p = \langle \gamma - 1 \rangle \left[e - \frac{1}{2} \rho \langle u^2 + v^2 + w^2 \rangle \right]$$
$$a^2 = \frac{\gamma p}{\rho} = \gamma \langle \gamma - 1 \rangle \left[\frac{e}{\rho} - \frac{1}{2} \langle u^2 + v^2 + w^2 \rangle \right] \qquad (2.12)$$

Again, we use Stokes' Hypothesis to define the bulk viscosity, λ, as shown in Eq. (2.6).

As in the 2-D derivation, the dependent variables are decomposed into steady and fluctuating components, and the equations are time-averaged. This gives rise to Reynolds

stresses, which are related to the rate of mean strain using Boussinesq's concept. With this, an effective viscosity is defined, which consists of the molecular viscosity and a computed eddy viscosity that represents the contribution of the turbulent stresses to the mean flow. More information on the eddy viscosity formulation is given in the section on numerical formulation.

The nondimensionalization has been carried out in an identical manner to the 2-D equations.

Transformation of the 2-D Equations to Curvilinear Coordinates

From the viewpoint of computational accuracy, a uniform Cartesian grid is the most desirable geometry. However, most problems of interest have bodies which are not easy to fit a Cartesian grid about. Accurate computation of physical flow features can also give rise to conflicting needs. For example, resolving the boundary layer requires a fine grid near the body. If a uniform grid spacing is used, this puts many more grid points than are needed in the inviscid flow region, which is farther away from the body. These unnecessary grid points increase computation time and memory immensely. For easy application of boundary conditions and to accurately capture the physics of the problem, it is desirable to have a grid which wraps around the airfoil and has many points near the body, but is stretched in order to put fewer grid points in regions where they are not necessary.

To resolve these conflicting needs, the governing equations are transformed into generalized curvilinear coordinates. This transformation maps a stretched, body-fitted grid in the physical plane (x,z,t) to a uniform Cartesian grid in the computational plane (ξ,ζ,τ). This transformation is one-to-one (each point in the physical plane has a corresponding

point in the computational plane), except along any cuts needed to make the physical plane simply connected.

The general transformation between the two planes is given as:

$$\begin{aligned} \xi &= \xi(x,z,t) \\ \zeta &= \zeta(x,z,t) \\ \tau &= t \end{aligned} \qquad (2.13)$$

which transforms the physical domain (x,y,t) to the computational domain (ξ,ζ,τ).

Using this transformation, the curvilinear flow equations can be obtained; the detailed derivation is given in Appendix A. After the transformation, Eq. (2.2) is written as:

$$\partial_\tau \mathbf{q} + \partial_\xi \mathbf{E} + \partial_\zeta \mathbf{G} = \frac{1}{\mathbf{Re}} \left\langle \partial_\xi \mathbf{R} + \partial_\zeta \mathbf{T} \right\rangle \qquad (2.14)$$

where:

$$q=\frac{1}{J}\begin{pmatrix}\rho\\ \rho u\\ \rho w\\ e\end{pmatrix};\ E=\frac{1}{J}\begin{pmatrix}\rho u\\ \rho uU+\xi_x p\\ \rho wU+\xi_z p\\ U\langle e+p\rangle-\xi_t p\end{pmatrix};\ G=\frac{1}{J}\begin{pmatrix}\rho w\\ \rho uW+\zeta_x p\\ \rho wW+\zeta_y p\\ W\langle e+p\rangle-\zeta_t p\end{pmatrix}$$

$$R=\frac{1}{J}\begin{pmatrix}0\\ \xi_x\tau_{xx}+\xi_z\tau_{xz}\\ \xi_x\tau_{xz}+\xi_z\tau_{zz}\\ \xi_x R_4+\xi_z T_4\end{pmatrix};\ T=\frac{1}{J}\begin{pmatrix}0\\ \zeta_x\tau_{xx}+\zeta_z\tau_{xz}\\ \zeta_x\tau_{xz}+\zeta_z\tau_{zz}\\ \zeta_x R_4+\zeta_z T_4\end{pmatrix} \qquad (2.15)$$

with the contravariant velocities U and W given as:

$$U=\xi_t+\xi_x u+\xi_z w$$
$$W=\zeta_t+\zeta_x u+\zeta_z w \qquad (2.16)$$

The Jacobian of transformation is defined as:

$$J=\xi_x\zeta_z-\xi_z\zeta_x=\frac{1}{x_\xi z_\zeta-x_\zeta z_\xi} \qquad (2.17)$$

The transformed viscous terms become:

$$\tau_{xx} = \mu\left[\frac{4}{3}\left(\xi_x u_\xi + \zeta_x u_\zeta\right) - \frac{2}{3}\left(\xi_z w_\xi + \zeta_z w_\zeta\right)\right]$$

$$\tau_{xz} = \mu\left[\left(\xi_z u_\xi + \zeta_z u_\zeta\right) + \left(\xi_x w_\xi + \zeta_x w_\zeta\right)\right]$$

$$\tau_{zz} = \mu\left[\frac{4}{3}\left(\xi_z w_\xi + \zeta_z w_\zeta\right) - \frac{2}{3}\left(\xi_x u_\xi + \zeta_x u_\zeta\right)\right]$$

$$R_4 = u\tau_{xx} + w\tau_{xz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_x \partial_\xi a^2 + \zeta_x \partial_\zeta a^2\right)$$

$$T_4 = u\tau_{xz} + w\tau_{zz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_z \partial_\xi a^2 + \zeta_z \partial_\zeta a^2\right) \qquad (2.18)$$

The metric quantities can be related to the physical quantities by these relations:

$$\xi_x = Jz_\zeta$$

$$\xi_z = -Jx_\zeta$$

$$\xi_t = -x_\tau \xi_x - z_\tau \xi_z$$

$$\zeta_x = -Jz_\xi$$

$$\zeta_z = Jx_\xi$$

$$\zeta_t = -x_\tau \zeta_x - z_\tau \zeta_z \qquad (2.19)$$

Transformation of the 3-D Equations to Curvilinear Coordinates

The 3-D Navier-Stokes equations are transformed into curvilinear coordinates in a similar manner. This time, the general transformation is given as:

$$\xi = \xi(x,y,z,t)$$

$$\eta = \eta(x,y,z,t)$$

$$\zeta = \zeta(x,y,z,t)$$

$$\tau = t \qquad (2.20)$$

which transforms the physical domain (x,y,z,t) to the computational domain (ξ,η,ζ,τ).

Using this transformation, the curvilinear flow equations can be obtained; the detailed derivation follows that given in Appendix A. After the transformation, Eq. (2.9) is written as:

$$\partial_\tau \mathbf{q} + \partial_\xi \mathbf{E} + \partial_\eta \mathbf{F} + \partial_\zeta \mathbf{G} = \frac{1}{\mathbf{Re}} \left\langle \partial_\xi \mathbf{R} + \partial_\eta \mathbf{S} + \partial_\zeta \mathbf{T} \right\rangle \qquad (2.21)$$

where:

$$\mathbf{q} = \frac{1}{J} \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{pmatrix}; \quad \mathbf{E} = \frac{1}{J} \begin{pmatrix} \rho u \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U\langle e+p \rangle - \xi_t p \end{pmatrix}; \quad \mathbf{F} = \frac{1}{J} \begin{pmatrix} \rho v \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V\langle e+p \rangle - \eta_t p \end{pmatrix};$$

$$\mathbf{G}=\frac{1}{J}\begin{pmatrix}\rho w\\ \rho u W+\zeta_x p\\ \rho v W+\zeta_y p\\ \rho w W+\zeta_z p\\ W\langle e+p\rangle-\zeta_t p\end{pmatrix};\ \mathbf{R}=\frac{1}{J}\begin{pmatrix}0\\ \xi_x\tau_{xx}+\xi_y\tau_{xy}+\xi_z\tau_{xz}\\ \xi_x\tau_{xy}+\xi_y\tau_{yy}+\xi_z\tau_{yz}\\ \xi_x\tau_{xz}+\xi_y\tau_{yz}+\xi_z\tau_{zz}\\ \xi_x R_5+\xi_y S_5+\xi_z T_5\end{pmatrix};$$

$$\mathbf{S}=\frac{1}{J}\begin{pmatrix}0\\ \eta_x\tau_{xx}+\eta_y\tau_{xy}+\eta_z\tau_{xz}\\ \eta_x\tau_{xy}+\eta_y\tau_{yy}+\eta_z\tau_{yz}\\ \eta_x\tau_{xz}+\eta_y\tau_{yz}+\eta_z\tau_{zz}\\ \eta_x R_5+\eta_y S_5+\eta_z T_5\end{pmatrix};\ \mathbf{T}=\frac{1}{J}\begin{pmatrix}0\\ \zeta_x\tau_{xx}+\zeta_y\tau_{xy}+\zeta_z\tau_{xz}\\ \zeta_x\tau_{xy}+\zeta_y\tau_{yy}+\zeta_z\tau_{yz}\\ \zeta_x\tau_{xz}+\zeta_y\tau_{yz}+\zeta_z\tau_{zz}\\ \zeta_x R_5+\zeta_y S_5+\zeta_z T_5\end{pmatrix} \tag{2.22}$$

with the contravariant velocities U, V, and W given as:

$$\begin{aligned}U&=\xi_t+\xi_x u+\xi_y v+\xi_z w\\ V&=\eta_t+\eta_x u+\eta_y v+\eta_z w\\ W&=\zeta_t+\zeta_x u+\zeta_y v+\zeta_z w\end{aligned} \tag{2.23}$$

The Jacobian of transformation is defined as:

$$J = \xi_x\left(\eta_y\zeta_z - \eta_z\zeta_y\right) + \xi_y\left(\eta_z\zeta_x - \eta_x\zeta_z\right) + \xi_z\left(\eta_x\zeta_y - \eta_y\zeta_x\right)$$
$$= \frac{1}{y_\xi\left(x_\zeta z_\eta - x_\eta z_\zeta\right) + y_\eta\left(x_\xi z_\zeta - x_\zeta z_\xi\right) + y_\zeta\left(x_\eta z_\xi - x_\xi z_\eta\right)} \qquad (2.24)$$

The transformed viscous terms become:

$$\tau_{xx} = \mu\left[\frac{4}{3}\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta\right) - \frac{2}{3}\left(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta\right) - \frac{2}{3}\left(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right)\right]$$

$$\tau_{xy} = \mu\left[\left(\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta\right) + \left(\xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta\right)\right]$$

$$\tau_{xz} = \mu\left[\left(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta\right) + \left(\xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta\right)\right]$$

$$\tau_{yy} = \mu\left[\frac{4}{3}\left(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta\right) - \frac{2}{3}\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta\right) - \frac{2}{3}\left(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right)\right]$$

$$\tau_{yz} = \mu\left[\left(\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta\right) + \left(\xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta\right)\right]$$

$$\tau_{zz} = \mu\left[\frac{4}{3}\left(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right) - \frac{2}{3}\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta\right) - \frac{2}{3}\left(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta\right)\right]$$

$$R_5 = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_x\partial_\xi a^2 + \eta_x\partial_\eta a^2 + \zeta_x\partial_\zeta a^2\right)$$

$$S_5 = u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_y\partial_\xi a^2 + \eta_y\partial_\eta a^2 + \zeta_y\partial_\zeta a^2\right)$$

$$T_5 = u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_z\partial_\xi a^2 + \eta_z\partial_\eta a^2 + \zeta_z\partial_\zeta a^2\right) \qquad (2.25)$$

The metric quantities can be related to the physical quantities by these relations:

$$
\begin{aligned}
\xi_x &= J(y_\eta z_\zeta - y_\zeta z_\eta) \\
\xi_y &= J(x_\zeta z_\eta - x_\eta z_\zeta) \\
\xi_z &= J(x_\eta y_\zeta - y_\eta x_\zeta) \\
\eta_x &= J(z_\xi y_\zeta - y_\xi z_\zeta) \\
\eta_y &= J(x_\xi z_\zeta - x_\zeta z_\xi) \\
\eta_z &= J(y_\xi x_\zeta - x_\xi y_\zeta) \\
\zeta_x &= J(y_\xi z_\eta - z_\xi y_\eta) \\
\zeta_y &= J(z_\xi x_\eta - x_\xi z_\eta) \\
\zeta_z &= J(x_\xi y_\eta - y_\xi x_\eta) \\
\xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z \\
\eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z \\
\zeta_t &= -x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z
\end{aligned}
\qquad (2.26)
$$

CHAPTER III

NUMERICAL FORMULATION

In this chapter, two time-accurate finite difference schemes are described for numerically integrating the equations given in the previous chapter. One formulation discussed uses an Alternating Direction Implicit (ADI) Newton iteration scheme at each time step, while the other uses an LU-SGS scheme with Newton iteration. Then, the GMRES formulation is explained, and several variations of the method are described. To simplify the derivation, only the two dimensional equations are considered; the differences between the 2-D and 3-D schemes are mentioned as they are reached.

Iterative ADI formulation

The underlying code in the GMRES formulation is a Newton iteration ADI solver. This code is used as a function evaluator for the GMRES, as described in the next section. A brief outline of the ADI Newton algorithm is given below.

Discretization in Time and Space

The 2-D Reynolds averaged Navier-Stokes equations written in curvilinear form are given as:

$$\partial_\tau \mathbf{q} + \partial_\xi \mathbf{E} + \partial_\zeta \mathbf{G} = \frac{1}{Re}\left\langle \partial_\xi \mathbf{R} + \partial_\zeta \mathbf{T} \right\rangle \tag{3.1}$$

This equation is discretized by using the Euler implicit scheme, which is first order accurate in time and second order accurate in space. The time derivative is approximated by a first order forward difference, while the spatial derivatives are represented by second order central differences. Using Taylor series expansions, Eq. (3.1) can be rewritten:

$$\frac{\left(q^{n+1,k+1} - q^{n+1,k}\right)}{\Delta\tau} + \left(\delta_\xi E^{n+1,k+1} + \delta_\zeta G^{n+1,k+1}\right)$$
$$= \frac{-\left(q^{n+1,k} - q^{n}\right)}{\Delta\tau} + \frac{1}{Re}\left(\delta_\xi R^{n+1,k} + \delta_\zeta T^{n+1,k}\right) + O\left(\Delta\tau, \Delta\xi^2, \Delta\zeta^2\right) \tag{3.2}$$

where $O(\Delta\tau,\Delta\xi^2,\Delta\zeta^2)$ indicates that this expression is first order accurate in time (second order terms are truncated), and second order accurate in space. In Eq. (3.2), 'n' refers to the time level and 'k' refers to the Newton iteration level at that time step. The details of this derivation are given in Appendix B.

Linearization of the Governing Equation

Given the known flow variables at the 'n' time level and a previous guess for the flow variables at the 'n+1' time level, equation set (3.2) can now be iterated upon to obtain the flow variables at the 'n+1' time level. Unfortunately, this set of algebraic equations are coupled and highly nonlinear, making them very difficult to solve. To make these equations easier to solve, the convection terms **E** and **G** are linearized about time level

'n+1' and iteration level 'k' by means of Taylor series. The linearization procedure is described in detail in Appendix C. When this is substituted into (3.2), the linearized equations are written as:

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k} + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} =$$
$$-\Delta\tau\left(\frac{q^{n+1,k} - q^{n}}{\Delta\tau}\right) - \Delta\tau\left\{\delta_\xi E^{n+1,k} + \delta_\zeta G^{n+1,k}\right\} + \frac{\Delta\tau}{Re}\left\{\delta_\xi R^{n+1,k} + \delta_\zeta T^{n+1,k}\right\} \tag{3.3}$$

This equation set is first order accurate in time and second order accurate in space. The matrix to be solved is in block pentadiagonal form.

Approximate Factorization of the Governing Equation

Equation (3.3) is a large, sparse pentadiagonal block matrix equation. This is still very expensive to solve, requiring large amounts of storage and computation. Instead of solving Eq. (3.3) directly, it is factored into a series of one dimensional block tridiagonal systems of equations, using the approximate factorization technique of Beam and Warming (Ref. 18).

In this method, the left hand side of Eq. (3.3) is approximately factored into two operators:

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k}\right\}\left\{I + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} =$$
$$\Delta\tau\left\{RHS^{n+1,k}\right\} - \Delta\tau^2\delta_\xi A^{n+1,k}\delta_\zeta C^{n+1,k}\Delta q^{n+1,k} \tag{3.4}$$

where

$$\left\{RHS^{n+1,k}\right\} =$$

$$-\frac{q^{n+1,k} - q^{n}}{\Delta\tau} - \left(\delta_\xi E^{n+1,k} + \delta_\zeta G^{n+1,k}\right) + \frac{1}{Re}\left(\delta_\xi R^{n+1,k} + \delta_\zeta T^{n+1,k}\right) \quad (3.5)$$

The last term on the right hand side of Eq. (3.4) is second order in time, and can thus be dropped without degrading the formal first order time accuracy of the scheme. This gives the factored set of equations to be solved:

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k}\right\}\left\{I + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\} \quad (3.6)$$

Details of the solution procedure are given in Appendix D.

In solving Eq. (3.6) for subsonic and transonic flows, it is necessary to add artificial viscosity to damp the numerical oscillations. The numerical viscosity model proposed by Jameson, Turkel, and Schmidt, and modified by Swanson and Turkel (Ref. 19) is used. The details of this model are given in Appendix E.

When viscous flows at high Reynolds numbers are solved, it becomes necessary to consider turbulent effects. While the present equations can directly model turbulent motion, the small time step and dense grid that is required for accuracy make the computational cost prohibitive. To keep a reasonable grid spacing, Eq. (3.3) is time-averaged and the well-known Baldwin-Lomax algebraic turbulence model is employed to represent the turbulent stresses. The details of the model are given in Appendix F.

Finally, to solve Eq. (3.6), initial and boundary conditions are required. Appendix G describes these conditions and their implementation.

LU-SGS Formulation

The LU-SGS solver was only implemented in the 3-D GMRES code. However, to save space, the 2-D version of the algorithm will be described.

To derive the numerical formulation, we start with the linearized form of the 2-D governing equations, Eq. (3.3):

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k} + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\} \quad (3.3)$$

The RHS term is defined in Eq. (3.5). The left hand side of Eq. (3.3) can be factorized in this manner:

$$\left\{I + \theta\Delta\tau\left(\delta_\xi^- A^+ + \delta_\zeta^- C^+ + \delta_\xi^+ A^- + \delta_\zeta^+ C^-\right)^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\} \quad (3.7)$$

where θ is a user-defined scalar (>0.5). Also, δ^+ represents a forward difference, and δ^- represents a backward difference. For example:

$$\delta_\xi^- A = \frac{A_i - A_{i-1}}{\Delta\xi}$$
$$\delta_\xi^+ A = \frac{A_{i+1} - A_i}{\Delta\xi} \quad (3.8)$$

In Eq. (3.7), $\mathbf{A}^+$, $\mathbf{A}^-$, $\mathbf{B}^+$, and $\mathbf{B}^-$ are constructed so that (+) matrices have positive eigenvalues, and (-) matrices have negative eigenvalues. The definition of these matrices are very important to the success of the LU method. In this formulation, the following definitions are used, from Ref. 23:

$$\begin{aligned}
\mathbf{A}^+ &= \frac{1}{2}\left(\mathbf{A} + \beta\lambda_A\mathbf{I}\right) \\
\mathbf{A}^- &= \frac{1}{2}\left(\mathbf{A} - \beta\lambda_A\mathbf{I}\right) \\
\mathbf{C}^+ &= \frac{1}{2}\left(\mathbf{C} + \beta\lambda_C\mathbf{I}\right) \\
\mathbf{C}^- &= \frac{1}{2}\left(\mathbf{C} - \beta\lambda_C\mathbf{I}\right)
\end{aligned} \tag{3.9}$$

where β is a user-defined parameter, and:

$$\begin{aligned}
\lambda_A &= |U| + a\sqrt{\xi_x^2 + \xi_z^2} \\
\lambda_C &= |W| + a\sqrt{\zeta_x^2 + \zeta_z^2}
\end{aligned} \quad \text{(2-D)}$$

$$\begin{aligned}
\lambda_A &= |U| + a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2} \\
\lambda_B &= |V| + a\sqrt{\eta_x^2 + \eta_y^2 + \eta_z^2} \\
\lambda_C &= |W| + a\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2}
\end{aligned} \quad \text{(3-D)} \tag{3.10}$$

These are the eigenvalues of the **A** and **C** matrices.

Following the development given in Ref. 23, Eq. (3.7) can be written in conservative form as:

$$\Delta q_{i,j}^{n+1,k} + \theta\Delta\tau \left(\begin{array}{l} \left[A_{i,j}^{+} \Delta q_{i,j}^{n+1,k} - A_{i-1,j}^{+} \Delta q_{i-1,j}^{n+1,k} \right] + \\ \left[A_{i+1,j}^{-} \Delta q_{i+1,j}^{n+1,k} - A_{i,j}^{-} \Delta q_{i,j}^{n+1,k} \right] + \\ \left[C_{i,j}^{+} \Delta q_{i,j}^{n+1,k} - C_{i,j-1}^{+} \Delta q_{i,j-1}^{n+1,k} \right] + \\ \left[C_{i,j+1}^{-} \Delta q_{i,j+1}^{n+1,k} - C_{i,j}^{-} \Delta q_{i,j}^{n+1,k} \right] \end{array} \right) = \Delta\tau \left\{ RHS_{i,j}^{n+1,k} \right\} \tag{3.11}$$

For convenience, the superscript 'n+1,k' is omitted for the rest of this development.

Next, Eq. (3.11) is simulated using forward and backward sweeps, which is written as:

$$\Delta q_{i,j}^{*} + \theta\Delta\tau \left(\begin{array}{l} \Delta q_{i,j}^{*} \left[A_{i,j}^{+} - A_{i,j}^{-} \right] + \\ A_{i+1,j}^{-} \Delta q_{i+1,j}^{*} + \\ \Delta q_{i,j}^{*} \left[C_{i,j}^{+} - C_{i,j}^{-} \right] + \\ C_{i,j+1}^{-} \Delta q_{i,j+1}^{*} \end{array} \right) = \Delta\tau \left\{ RHS_{i,j}^{n+1,k} \right\} \tag{3.12}$$

and

$$\Delta q_{i,j}^{n+1,k} + \theta\Delta\tau \left(\begin{array}{l} \Delta q_{i,j}^{n+1,k} \left[A_{i,j}^{+} - A_{i,j}^{-} \right] - \\ A_{i-1,j}^{+} \Delta q_{i-1,j}^{n+1,k} + A_{i+1,j}^{-} \Delta q_{i+1,j}^{*} + \\ \Delta q_{i,j}^{n+1,k} \left[C_{i,j}^{+} - C_{i,j}^{-} \right] - \\ C_{i,j-1}^{+} \Delta q_{i,j-1}^{n+1,k} + C_{i,j+1}^{-} \Delta q_{i,j+1}^{*} \end{array} \right) = \Delta\tau \left\{ RHS_{i,j}^{n+1,k} \right\} \tag{3.13}$$

If Eq. (3.12) is subtracted from Eq. (3.13), the following is obtained:

$$\Delta q_{i,j}^{n+1,k} + \theta\Delta\tau \begin{pmatrix} \Delta q_{i,j}^{n+1,k}\left[A_{i,j}^{+} - A_{i,j}^{-}\right] - \\ A_{i-1,j}^{+}\Delta q_{i-1,j}^{n+1,k} + \\ \Delta q_{i,j}^{n+1,k}\left[C_{i,j}^{+} - C_{i,j}^{-}\right] - \\ C_{i,j-1}^{+}\Delta q_{i,j-1}^{n+1,k} \end{pmatrix} =$$

$$\Delta q_{i,j}^{*} + \theta\Delta\tau \begin{pmatrix} \Delta q_{i,j}^{*}\left[A_{i,j}^{+} - A_{i,j}^{-}\right] + \\ \Delta q_{i,j}^{*}\left[C_{i,j}^{+} - C_{i,j}^{-}\right] \end{pmatrix} \tag{3.14}$$

This may be written as:

$$\left\{I + \theta\Delta\tau\left(\delta_{\xi}^{-}A^{+} + \delta_{\zeta}^{-}C^{+} - A^{-} - C^{-}\right)^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\}$$
$$= \left\{I + \theta\Delta\tau\left(A^{+} + C^{+} - A^{-} - C^{-}\right)^{n+1,k}\right\}\left\{\Delta q^{*}\right\} \tag{3.15}$$

where:

$$\left\{I + \theta\Delta\tau\left(\delta_{\xi}^{+}A^{-} + \delta_{\zeta}^{+}C^{-} + A^{+} + C^{+}\right)^{n+1,k}\right\}\left\{\Delta q^{*}\right\}$$
$$= \Delta\tau\left\{RHS^{n+1,k}\right\} \tag{3.16}$$

If we use the plus and minus matrices defined above in Eq. (3.10), then Eq. (3.15) becomes:

$$\left\{I + \theta\Delta\tau\left(\delta_\xi^- A^+ + \delta_\zeta^- C^+ - A^- - C^-\right)^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\}$$
$$= \left\{I + \theta\Delta\tau\left(\lambda_A I + \lambda_C I\right)^{n+1,k}\right\}\left\{\Delta q^*\right\} \qquad (3.17)$$

Equations (3.16) and (3.17) together define the LU-SGS procedure. In order to change Eq. (3.16) and (3.17) into a Newton-like iteration, the time step is increased to infinity and θ is set equal to one. This removes the requirement for numerical investigation to determine an optimal time step. The Newton-like iterative procedure is written as:

$$\left\{\left(\delta_\xi^+ A^- + \delta_\zeta^+ C^- + A^+ + C^+\right)^{n+1,k}\right\}\left\{\Delta q^*\right\}$$
$$= \left\{RHS^{n+1,k}\right\}$$
$$\left\{\left(\delta_\xi^- A^+ + \delta_\zeta^- C^+ - A^- - C^-\right)^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\}$$
$$= \left\{\left(\lambda_A I + \lambda_C I\right)^{n+1,k}\right\}\left\{\Delta q^*\right\} \qquad (3.18)$$

This procedure is used for all LU-SGS computations described in this report.

GMRES Formulation

Description of GMRES algorithm

The iterative ADI or LU-SGS formulation given above may be expressed in this way:

$$\mathbf{q}^{n+1,k+1} = \mathbf{F}\left(\mathbf{q}^{n+1,k}\right) \tag{3.19}$$

Thus, given a guess for $\mathbf{q}^{n+1,k}$, the solver returns a (hopefully) better approximation, $\mathbf{q}^{n+1,k+1}$, to the correct solution. When the solution has converged (i.e., $\mathbf{q}^{n+1,k} = \mathbf{q}^{n+1,k+1}$), then:

$$\mathbf{F}\left(\mathbf{q}^{n+1,k}\right) - \mathbf{q}^{n+1,k} = \mathbf{M}\left(\mathbf{q}^{n+1,k}\right) = 0 \tag{3.20}$$

Eq. (3.19) may be rewritten as:

$$\begin{aligned} \mathbf{q}^{n+1,k+1} &= \mathbf{q}^{n+1,k} + \Delta\mathbf{q} \\ &= \mathbf{q}^{n+1,k} + \mathbf{a}\vec{\mathbf{d}} \end{aligned} \tag{3.21}$$

where:

$$\vec{\mathbf{d}} = \frac{\overrightarrow{\Delta\mathbf{q}}}{\|\Delta\mathbf{q}\|} \tag{3.22}$$

and

$$\mathbf{a} = \|\Delta\mathbf{q}\| \tag{3.23}$$

In words, using the latest guess for the solution at the new time level, the original code computes a corrected solution $\mathbf{q}^{n+1,k+1}$, which is equivalent to moving a distance **a** in direction **d** from the initial point $\mathbf{q}^{n+1,k}$.

In a two dimensional problem, the Δq vector has (imax * kmax * 4) entries. The correction vector may change only one flow variable at one point in the flow field (e.g., ρu at i=5, k=13), and leave the rest alone. This is one possible direction that the code could move in. If another variable at another point is changed instead (e.g., ρ at i=120, k=2), this would result in the code moving in a second direction which is orthogonal to the first. Thus it can be seen that there are a total of (imax * kmax * 4) possible orthogonal directions in a 2-D problem, and (imax * jmax * kmax * 5) directions in 3-D.

The ADI and LU-SGS iterative codes both consider only one direction at a time. In words, they start from an initial point, compute a single likely direction, and move some distance in this direction to the next point, where the same process is repeated.

The GMRES solver works in a different way. GMRES computes the slope of the residual in a number of orthogonal directions from the initial point, and uses this information to make a more informed move from the initial point. In this procedure, the underlying iterative solver serves as a 'black box' function evaluator (i.e., given a set of input flow properties, the solver sends back an updated set of flow properties) to provide GMRES with information to compute the set of flow properties that will satisfy Eq. (3.20).

Note that GMRES does not change with the number of equations or the method of solution of the underlying code. The only change in GMRES for 2-D to 3-D is the length of the vectors; there is no change in the GMRES code between the ADI and LU-SGS solvers.

A GMRES step follows this procedure:

First, the initial direction is computed as

$$\vec{d}_1 = M(q^{n+1,k}) \qquad (3.24)$$

and normalized as

$$\vec{d}_1 = \frac{\vec{d}_1}{\|\vec{d}_1\|} \tag{3.25}$$

Thus the first direction is the direction in which the underlying solver would have moved from the initial point.

To compute the remaining search directions (j=1,2,..,J-1), the GMRES solver first moves a small distance in the j^{th} direction and calls the underlying solver in order to compute the residual at this point. Then, the slope of the residual in the j^{th} direction can be numerically evaluated using:

$$\overline{M}(q\,;\vec{d}) = \frac{M(q + \varepsilon\vec{d}) - M(q)}{\varepsilon} \tag{3.26}$$

where ε is taken to be some small number. In this work, ε is taken to be 0.001.

Taking the dot product of this derivative with a unit direction vector will reveal the component of the derivative in that direction:

$$b_{ij} = (\overline{M}(q^{n+1,k}\,;\vec{d}_j), \vec{d}_i) \tag{3.27}$$

If the components of the derivative in all of the known directions are subtracted from the derivative, what results is a new direction vector that is orthogonal to all of the known directions:

$$\vec{d}_{j+1} = \overline{M}(q^{n+1,k} ; \vec{d}_j) - \sum_{i=1}^{j} b_{ij} \vec{d}_i \qquad (3.28)$$

Normalizing the new direction vector will give the component of the derivative in the new direction:

$$b_{j+1,j} = \left\| \vec{d}_{j+1} \right\| \qquad (3.29)$$

and the unit vector in the new direction may finally be computed as:

$$\vec{d}_{j+1} = \frac{\vec{d}_{j+1}}{b_{j+1,j}} \qquad (3.30)$$

Since GMRES uses the underlying flow solver to determine the search directions, the success and speed of the GMRES solution method depends greatly on the original flow solver's ability to help define useful direction vectors, and hence a subspace that contains much of the error components.

After obtaining the search directions, the solution vector is updated using

$$q^{n+1,k+1} = q^{n+1,k} + \sum_{j=1}^{J} a_j \vec{d}_j \qquad (3.31)$$

where the undetermined coefficients a_j are chosen to minimize:

$$\left\|M(q^{n+1,k+1})\right\|^2 = \left\| M(q^{n+1,k} + \sum_{j=1}^{J} a_j \vec{d}_j) \right\|^2$$
$$\approx \left\| M(q^{n+1,k}) + \sum_{j=1}^{J} a_j \overline{M}(q^{n+1,k}; \vec{d}_j) \right\|^2 . \qquad (3.32)$$

This equation is minimized as follows:

Let D_j be the matrix of directions $\{d_1, d_2, d_3, ..., d_j\}$.. Also, let F_j be the matrix of directional derivatives given as $\{M_1, M_2, M_3, ..., M_j\}$, where:

$$M_j = \overline{M}(q^{n+1,k}; \vec{d}_j) \qquad (3.33)$$

Then Eq. (3.28) may be rewritten in matrix form as:

$$M_j = D_{j+1} B \qquad (3.34)$$

Here, B is the (J+1) x (J) matrix:

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & & & b_{1,J-2} & b_{1,J-1} & b_{1,J} \\ b_{2,1} & b_{2,2} & b_{2,3} & & \cdots & b_{2,J-2} & b_{2,J-1} & b_{2,J} \\ 0 & b_{3,2} & b_{3,3} & & & b_{3,J-2} & b_{3,J-1} & b_{3,J} \\ & 0 & & & \ddots & & \vdots & \\ & & & & & b_{J-1,J-2} & b_{J-1,J-1} & b_{J-1,J} \\ & 0 & & & 0 & 0 & b_{J,J-1} & b_{J,J} \\ & & & & & 0 & 0 & b_{J+1,J} \end{bmatrix} \qquad (3.35)$$

Note that at this point, $b_{J+1,J}$ is not yet known. Saad and Schultz give the following formula for evaluating this term without another function evaluation:

$$b_{J+1,J}^2 = \left\| \overline{M}(q^{n+1,k}; \vec{d}_J) \right\|^2 - \sum_{i=1}^{J} b_{i,J}^2 \qquad (3.36)$$

At this point, Eq. (3.32) is rewritten:

$$\left\| M(q^{n+1,k}) + \sum_{j=1}^{J} a_j \overline{M}(q^{n+1,k}; \vec{d}_j) \right\|^2 = \left\| M(q^{n+1,k}) + M_j A \right\|^2 \qquad (3.37)$$

where A is the vector $\{a_1, a_2, a_3, ..., a_j\}^T$. Then, using the definition of the first direction and Eq. (3.34), Eq. (3.37) becomes:

$$\begin{aligned} &\left\| M(q^{n+1,k}) + M_j A \right\|^2 \\ &= \left\| \left(\left\| \vec{d}_1 \right\| \vec{d}_1 + M_j A \right) \right\|^2 \\ &= \left\| \left(\left\| \vec{d}_1 \right\| \vec{d}_1 + D_{j+1} B A \right) \right\|^2 \\ &= \left\| D_{j+1} \left(\left\| \vec{d}_1 \right\| e + B A \right) \right\|^2 \\ &= \left\| \left(\left\| \vec{d}_1 \right\| e + B A \right) \right\|^2 \end{aligned} \qquad (3.38)$$

where e is the first column of the (JxJ) identity matrix.

This least squares problem is solved using the QR algorithm in LINPACK.

Residual Definition for GMRES

It is important to remember here that GMRES is a completely separate routine from the rest of the code. GMRES is designed to minimize a given residual, which is calculated by the underlying code. This means that GMRES does not necessarily follow a physically meaningful path to the correct answer. For example, given a steady flow problem, the original code (with the Newton iteration disabled, because time-accuracy is not required) will march in time until a steady flow is obtained. On the way to the answer, the flow field at each time level, while not necessarily representing the real answer at that time level (especially for local time stepping), will follow the physics of the problem. GMRES, on the other hand, is given this definition for the residual:

$$\begin{aligned} \mathbf{M}(\mathbf{q}^n) &= \mathbf{q}^{n+1} - \mathbf{q}^n \\ &= \mathbf{F}(\mathbf{q}^n) - \mathbf{q}^n \end{aligned} \tag{3.39}$$

where $\mathbf{F}(\mathbf{q})$ is the result given by the underlying solver given an input $\mathbf{q}$. GMRES will get the same answer as the pseudo-time-marching code (in a hopefully shorter time), but it will do this by simply trying to drive the residual to zero as quickly as possible instead of following the physics of the flow.

Thus, with GMRES, it is important to define a level of residual where there is confidence in the answer. This is especially true in the 2-D inviscid transonic calculations that were performed. In these runs, the GMRES solver would attain a certain level of convergence, depending on the number of directions employed, and then stop converging. It was hypothesized that the GMRES solver found local minima in the residual that it could

not escape from without having more directions. This was supported by the behavior of the solver as directions were added: a lower residual was obtained, and a better solution was also calculated.

The solution, however, varied by up to 20% depending on the number of directions used (and thus the level of convergence reached). An investigation into the residual levels necessary for reliable answers is needed.

In these investigations, several residuals were defined for GMRES. In steady flow cases, Eq. (3.39) was used, while in unsteady calculations, Eq. (3.20) was employed.

Additional Techniques Employed with GMRES

The GMRES routine on its own gives very satisfactory convergence properties. The major drawback to this method is the amount of memory that is required: for an 'N' direction iteration, 'N' complete flow fields must be stored. In two dimensions, this is not too much of a problem; but when the code is extended to three dimensions, the memory required quickly becomes horrendous. Therefore, several methods were tried in the 2-D and 3-D codes to cut down the amount of memory required for the GMRES iteration. These are detailed below.

Newton iteration on GMRES at Each Time Step (Restart).

In an attempt to cut down the memory necessary to run GMRES, a Newton iteration was used at each time step on the GMRES evaluation. This was a practical way of cutting the memory in half. For example, instead of using one GMRES(10) iteration (a 10 direction GMRES iteration) at each time step (and storing the equivalent of 10 complete flow fields), two GMRES(5) iterations (storing only 5 flow fields) were performed. In this method, the first GMRES(5) iteration gives an updated $\mathbf{q}$ solution for the 'n+1' time

level, and this is used as the input guess for the next iteration (which is still at the 'n+1' time level). It was found that the residuals for this 'restart' method were equivalent or better than the residuals for the single evaluation method. A flow chart of a time step follows:

1) Start with a guess for $\mathbf{q}^{n+1}$: $\mathbf{q}^{n+1,0}$.

2) Perform a GMRES(5) iteration on $\mathbf{q}^{n+1,0}$ to get $\mathbf{q}^{n+1,1}$.

3) Use $\mathbf{q}^{n+1,1}$ as the input for another GMRES(5) iteration to get $\mathbf{q}^{n+1,2}$.

4) Go to the next time step.

This method trades more CPU time for less memory. Note that at time levels with smooth, attached flow, the second GMRES(5) iteration may not be necessary. To take advantage of this, a user-defined residual tolerance was implemented which allowed the second iteration to be skipped when the residual from the first iteration was low enough.

Multigrid GMRES iteration (2-D)

In order to further understand the GMRES procedure, several variables were investigated. First, the weighting coefficients of the first 20 directions during a dynamic stall loop were plotted. This showed the relative importance of the directions in lowering the residual at each time step. It was seen that the first 10 directions were by far the most important, with the next 10 directions playing an ever-decreasing role in the solution (i.e., the coefficient for the 20th direction was usually two orders of magnitude smaller than that of the first direction). Only at a very few steps were the higher directions weighted more.

Next, the first 20 directions of a GMRES iteration on an airfoil in dynamic stall at a single time step were plotted. This showed that the first directions were very smooth,

while the higher directions looked more jagged and noisy. At this point, it was hypothesized that a multigrid method could help speed convergence. Since the first directions were smooth (low frequency error), it was thought that a coarse grid evaluation with few directions could reduce this error more quickly.

The multigrid method worked as follows: a V pattern was adopted using two grid levels, with a GMRES(5) evaluation at each step in the pattern. On the fine grid, the residuals were defined as before (Eq. (3.36)), while the coarse grid used a Full Approximation Storage scheme residual:

$$M(q)_c = F(q)_c - q_c + M(q^1)_{f\to c} - M(q^1)_c \qquad (3.40)$$

where the subscript **c** denotes an evaluation on the coarse grid, and the subscript **f->c** refers to a variable transferred from the fine grid to the coarse grid. The additional last two terms in Eq. (3.40) are the residual from the first fine grid GMRES(5) evaluation. The first of these terms is the residual from the fine grid transferred to the coarse grid, while the second term is the residual evaluated on the coarse grid. These last two terms help to reduce any errors in the correction vector due to the grid switch.

At each time step, the following procedure is performed: first, a GMRES(5) iteration is done on the fine grid. The flow variables are updated and dropped to the coarse grid by averaging the fine grid variable values across the volume of the coarse grid. Then, a coarse grid GMRES(5) iteration is performed, using Eq. (3.40). At the end of the coarse grid evaluation, the correction vector is transferred back to the fine grid using bilinear interpolation, and is added to the fine grid flow variables. At this point, another GMRES(5) iteration is performed on the fine grid.

Parallel Steady Flow Analyses

After the GMRES code was investigated in 2-D on a sequential computer, it was ported to an Intel iPSC/860 parallel supercomputer for implementation in parallel. It was thought that with GMRES, communication bottlenecks in the distributed-memory algorithm may be reduced or eliminated.

Initially, the 2-D ADI code was reconfigured to run on an Intel iPSC/860. The Intel is a MIMD (multiple instruction/multiple data) machine; this means that the individual processors not only work with different subsets of the data, but each processor may independently execute different instructions at the same time.

The Intel is a distributed memory machine, where each individual processor has its own separate memory. In order to obtain information from another processor, a message-passing routine must be explicitly coded. Since message passing is very time-consuming on this machine, the most efficient code will usually have the least number of messages.

In order to accomplish this goal, the computational domain was divided up as shown in Figure (III.1). In this strategy, each subdomain overlaps its neighbors by two normal lines. Each processor performs an ADI step in its section of the domain; the overlapping lines are updated by passing messages at the end of each step.

The sequential version of the code is directly analogous to a single processor with a single computational domain. The main thrust of this work was to recover the sequential solution with a multiple processor machine with multiple computational domains.

Since an odd number of points is necessary to ensure that there is a leading edge point on the airfoil, the middle two processors both compute the leading edge line flow values. These are then averaged at the end of each step.

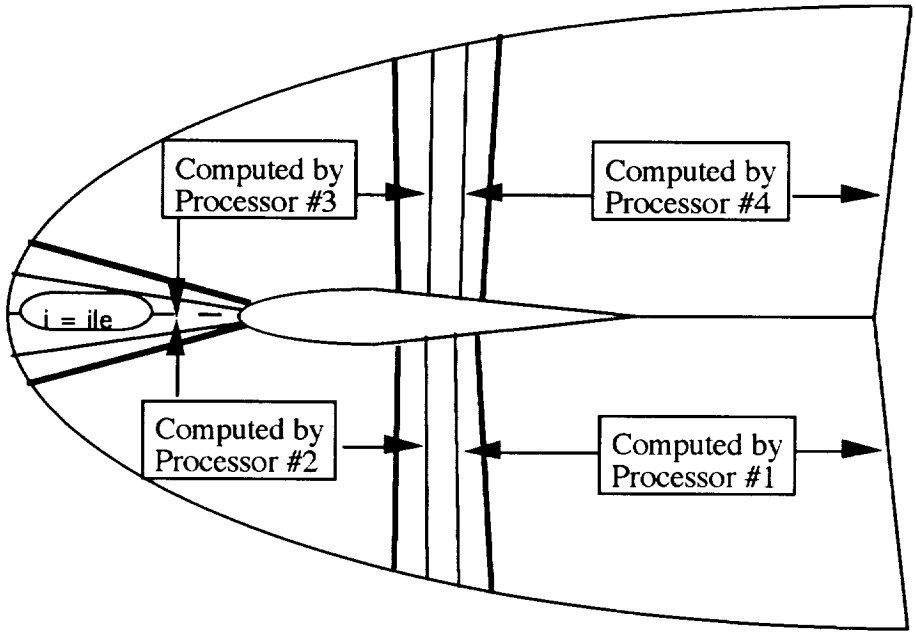

Figure (III.1):

Computational Domain Decomposition for a Four Processor MIMD Machine.

Iterative ADI Routine

It should be noted that a parallel ADI step is quite different than that of the sequential version. A description of the current implementation follows:

First, the order of the sweeps is reversed:

$$\left\{I + \Delta\tau\delta_\zeta \mathbf{C}^{n+1,k}\right\}\left\{I + \Delta\tau\delta_\xi \mathbf{A}^{n+1,k}\right\}\left\{\Delta\mathbf{q}^{n+1,k}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\}, \quad (3.41)$$

and the ζ sweep is performed. Since this sweep solves the k lines of the computational domain, line by line, the result is the same as for the sequential version:

$$\left\{I + \Delta\tau\delta_\zeta \mathbf{C}^{n+1,k}\right\}\left\{\Delta\mathbf{q}^{*}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\} \quad (3.42)$$

At this point, the sequential values for Δq^* are known throughout the flow field. Next, the second sweep is performed:

$$\left\{ I + \Delta\tau\delta_\xi A^{n+1,k} \right\} \left\{ \Delta q^{n+1,k} \right\} = \left\{ \Delta q^* \right\} \tag{3.43}$$

This sweep requires a bit more work to obtain an answer that is identical to the sequential solution. In the sequential code, which this is based on, the value of Δq is tacitly assumed to be zero at the boundaries (wall, freestream, and wake), as is the value of

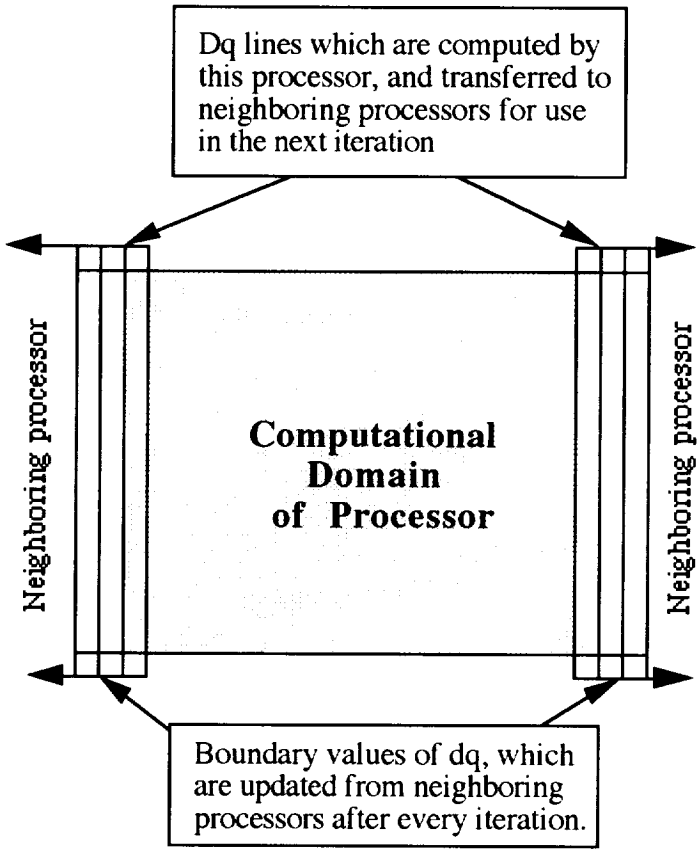

Figure (III.2): ξ Sweep Iteration Used for MIMD Computer

Δq^*. This is adequate for a single domain; however, when the flowfield is divided up between processors, it is seen that the value of Δq is obviously not zero on boundary lines.

To circumvent this problem, the ξ sweep is iterated upon, with the boundary values of Δq lagged one iteration and placed on the RHS of the equation. This procedure is shown in Figure (III.2). This requires two message passes per iteration for each interior processor, which increases run time dramatically. Also, the convergence rate of the boundary Δq values is not uniform between steps.

Block Cyclic Reduction Routine

To avoid the computation involved in the iterative ADI solver, a Block Cyclic Reduction (BCR) routine was implemented to directly solve the tridiagonal matrix in the ξ sweep. The BCR method is described below.

In an ADI method, the original single pentadiagonal matrix is approximately factored into two tridiagonal matrices. The sequential code used a Thomas algorithm to directly solve each matrix.

In a Thomas algorithm, the tridiagonal matrix is exactly factored into the product of a lower triangular matrix L and an upper triangular matrix U:

$$\begin{aligned} &\left[I + \Delta\tau\delta_\xi \mathbf{A}^{n+1,k}\right]\{\Delta q\} = \{R\} \\ \text{or: } &[M]\{\Delta q\} = \{R\} \\ \text{or: } &[L][U]\{\Delta q\} = \{R\} \end{aligned} \qquad (3.44)$$

Each entry in the tridiagonal matrix is itself a 4x4 matrix (2D) or a 5x5 matrix (3D). For simplicity, only a scalar tridiagonal matrix is shown; the extension to a block matrix is straightforward. In this way, Eq. (3.44) may be rewritten as:

$$\begin{bmatrix} b_1c_1 & 0 & 0 & 0 \\ a_2b_2c_2 & 0 & 0 & \\ 0\ a_3b_3c_3 & 0 & & \\ 0\ 0\ a_4b_4c_4 & & & \\ 0\ 0\ 0\ a_5b_5 & & & \end{bmatrix} = \begin{bmatrix} B_1 & 0 & 0 & 0 & 0 \\ A_2B_2 & 0 & 0 & 0 & \\ 0\ A_3B_3 & 0 & 0 & & \\ 0\ 0\ A_4B_4 & 0 & & & \\ 0\ 0\ 0\ A_5B_5 & & & & \end{bmatrix} \begin{bmatrix} 1 & C_1 & 0 & 0 & 0 \\ 0 & 1 & C_2 & 0 & 0 \\ 0 & 0 & 1 & C_3 & 0 \\ 0 & 0 & 0 & 1 & C_4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.45)$$

where:

$$\begin{aligned} A_j &= a_j \\ A_jC_{j-1} + B_j &= b_j \\ B_jC_j &= c_j \end{aligned} \quad (3.46)$$

Once the L and U matrices are known, then the tridiagonal matrix is solved in two steps. Eq. (3.44) is rewritten as:

$$\begin{aligned} [L][U]\{\Delta q\} &= \{R\} \\ &= [L]\{\Delta q^*\} \end{aligned} \quad (3.47)$$

and solved using:

$$\Delta q^*_j = \left(R_j - A_j \Delta q^*_{j-1}\right) / B_j \quad (3.48)$$

The second step is to solve:

$$[U]\{\Delta q\} = \{\Delta q^*\} \tag{3.49}$$

using:

$$\Delta q_j = \Delta q^*_j - C_j \Delta q_{j+1} \tag{3.50}$$

It is seen that the Thomas algorithm is inherently sequential; i.e., for each step in the inversion procedure, information is required from the step previously performed. Therefore, the Thomas algorithm is not a directly parallelizable procedure.

The Block Cyclic Reduction procedure is a more parallel way of directly solving the tridiagonal matrix. Given a tridiagonal matrix that is $((2^n+1) \times (2^n+1))$, the procedure works as follows.

First, the matrix is reduced:

$$\begin{bmatrix} b_1 c_1 & 0 & 0 & 0 \\ a_2 b_2 c_2 & 0 & 0 & \\ 0 & a_3 b_3 c_3 & 0 & \\ 0 & 0 & a_4 b_4 c_4 & \\ 0 & 0 & 0 & a_5 b_5 \end{bmatrix} = \begin{bmatrix} B_1 & 0 & C_1 & 0 & 0 \\ A_3 & 0 & B_3 & 0 & C_3 \\ 0 & 0 & A_5 & 0 & B_5 \end{bmatrix} \tag{3.51}$$

where:

$$
\begin{aligned}
A_j &= -\left(a_j a_{j-1}\right) / b_{j-1} \\
B_j &= b_j - \frac{\left(a_j c_{j-1}\right)}{b_{j-1}} - \frac{\left(c_j a_{j+1}\right)}{b_{j+1}} \\
C_j &= -\left(c_j c_{j+1}\right) / b_{j+1} \\
R_j &= r_j - \frac{\left(a_j r_{j-1}\right)}{b_{j-1}} - \frac{\left(c_j r_{j+1}\right)}{b_{j+1}}
\end{aligned}
\qquad (3.52)
$$

The reduction is continued, dropping every other line in each sweep, until only the center line remains:

$$B_j \Delta q_j = R_j \qquad (3.53)$$

At this point, the known Δq values are backsubstituted to find the unknown Δq values. The backsubstitution is performed in the reverse order as the reduction.

The BCR routine has three drawbacks. First, the processors must communicate before every round of reduction and backsubstitution to obtain matrix values that are outside of its computational domain. These messages are relatively short, however. Second, during the end of the reduction and the beginning of the backsubstitution when few lines are left to compute, there are processors idling. It is hoped that the savings in computation time compared to the parallel iterative routine will make up for the idle time encountered. Third, the BCR routine must have 2^n+1 lines to compute, which means that the grid required for the routine is much less flexible than that for the iterative parallel code. In order to meet this requirement, the number of i points in the grid was increased from 163 to 259.

It should be emphasized again that the BCR routine is a direct solver. Once through the BCR routine will return the correct answer to the tridiagonal matrix, unlike the iterative ADI solver described above.

GMRES Implementation

The GMRES routine was implemented on the parallel ADI code on a trial basis after the ADI code was validated. When GMRES was initially implemented, a question arose as to the definition of the residual to be minimized.

Two ideas were tried. The first idea was a completely parallel GMRES implementation, where each processor ran a GMRES routine to minimize the residual in its particular domain. When a function evaluation is required, the processors work together as before to compute a residual, but each processor is concerned only with the residual in its portion of the domain. This is equivalent to allowing each direction to have a different weighting coefficient in each processor's domain.

The second idea is a general GMRES implementation. In this scheme, the processors work as before to compute the search directions and the residual in its domain; but at the end of each function evaluation, the norm of the residual is globally computed and used. This is now equivalent to enforcing a single weighting coefficient to be used on each direction in the domain (i.e., as in the sequential version).

Due to time limitations, only steady flow applications were tested with the parallel code. The code supports time-accurate GMRES, however, and it is suggested that time-accurate tests be made.

The GMRES was also implemented on the BCR code, but due to memory limitations only 5 directions were used.

CHAPTER IV

2-D RESULTS

In 2-D, only the ADI code was used, since a satisfactory level of convergence was always obtainable by employing enough GMRES directions.

Validation of GMRES Code

Two steady-state cases were run with GMRES to validate it against the original ADI code. In all steady-state cases, a uniform time step was used for both GMRES and ADI.

From this point on, the term 'GMRES (J)' will be used to denote a J direction GMRES procedure used on a steady state flow problem.

The first case was inviscid transonic flow (Mach number of 0.8) over a NACA 0012 airfoil at a 1.25 degree angle of attack. This problem was chosen to see the effects of shocks on the GMRES solver. Figures 1 and 2 give the residual and lift coefficient history comparisons between the original ADI solver and the GMRES (40) code. The GMRES (40) solver requires only 50-55% of the CPU time necessary for the ADI code. Also, the lift coefficient converges much more rapidly.

The interesting part of this problem was in choosing the number of GMRES directions to use. A typical GMRES run would show the residual dropping rapidly at first, and then convergence would slow and stall. It was hypothesized that the GMRES procedure was getting caught in a local minima that it couldn't find a way out of. As more

directions were used in the GMRES procedure, the code converged to a lower level of residual before stalling, adding validity to this argument. When less than 40 directions were employed, the solution obtained would not match the result given by the original ADI code. With 40 directions, the result matched the ADI solution. A run of 80 directions showed that there was a limit to the speedup obtainable from using more directions; the convergence rate dropped sharply. It is thought that the higher directions contain much more noise than the early ones, and thus degrade the solution. Figure 3 shows a comparison of the global residuals for various GMRES runs.

One case was run with GMRES to validate it in the Navier-Stokes mode. The problem calculated was that of a NACA 0012 airfoil at a 5 degree angle of attack at M = 0.283 and a Reynolds number of 3,450,000.

Two GMRES runs were performed, with 10 and 40 directions used. Lift coefficient and residual histories are given in Figures 4 and 5, and the pressure distribution is compared to the ADI result in Figure 6. Excellent agreement is shown between the solvers.

The stalling phenomenon is seen again in Figure 5, but this time the level of residual obtained by the GMRES (10) solver was adequate for the problem.

Unsteady Flow Analyses

Once the code was validated, two test 2-D unsteady calculations were performed using the GMRES solver to determine if significant savings in CPU time may be obtained compared to the original ADI scheme.

It should be remembered that the approximate factorization used in deriving the original noniterative ADI code causes a factorization error to appear due to the splitting of the left hand side of the equations from one block pentadiagonal matrix to two tridiagonal

matrices. This error is proportional to the time step size. Therefore, to achieve time accuracy, the time step must be small enough to keep the factorization error negligible.

This limitation on the time step may be removed by performing a Newton iteration at each new time level in order to drive the factorization error to zero at each time step. Since each Newton iteration is equivalent to one ADI time step, the computation time is reduced if the time step multiplier **T** is greater than the number of iterations **N** required for an accurate solution at the new time level. This is illustrated in Figure (IV.1).

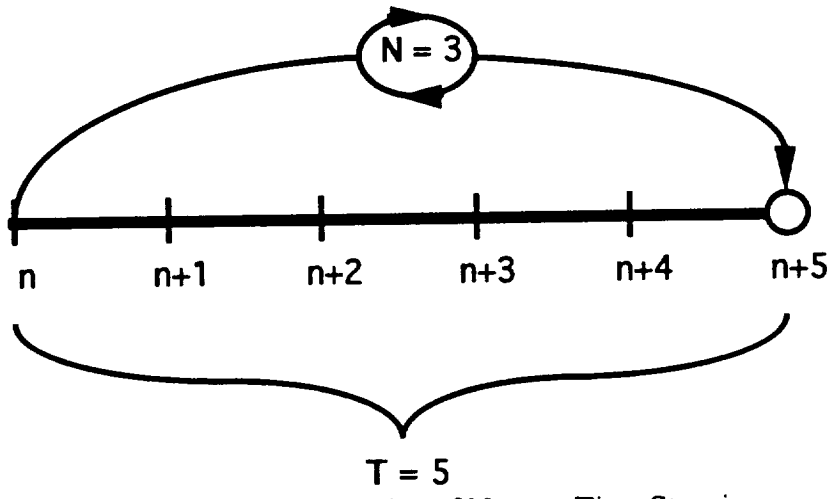

Figure (IV.1): Illustration of Newton Time Stepping

Note that the Newton iteration procedure is equivalent to a one direction GMRES step that is restarted **N** times.

The time step limitation may also be circumvented by using the correction vector from the Newton iterative solver as the function to be minimized by the GMRES solver. Since a **J** direction GMRES iteration requires **J** calls to the function evaluator (each of which is equivalent to an ADI time step), the computation time may be reduced in a similar manner to the Newton iterative procedure. This is illustrated in Figure (IV.2).

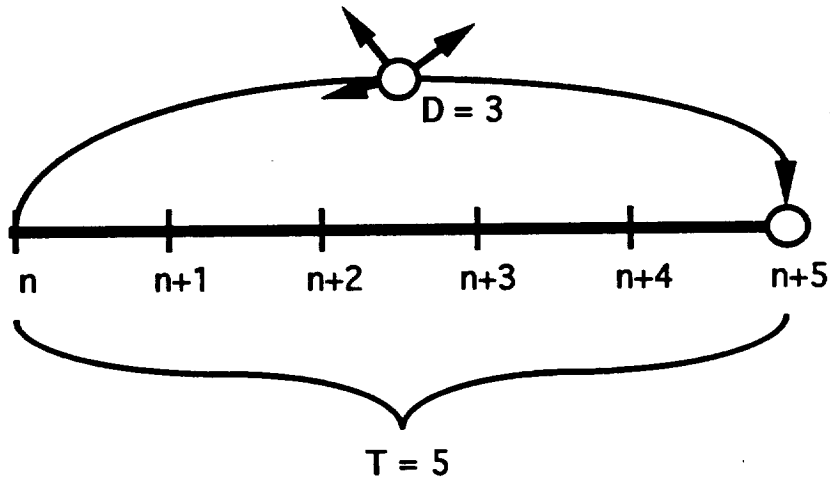

Figure (IV.2): Illustration of GMRES Time Stepping.

From this point on, the term 'GMRES (**J**/**T**)' will be used to denote a **J** direction GMRES procedure being performed at each time step of an unsteady calculation; the time step employed will be **T** times larger than the noniterative ADI time step.

The first test case evaluates the solver's ability to handle unsteady transonic flow. A plunging NACA 64A010 airfoil at a Mach number (M_∞) of 0.8 and a reduced frequency (based on half chord) of 0.2 is solved in the Euler mode. The plunging motion is defined by the equation

$$Y_\tau = -M_\infty \sin(1^o)\sin(\omega t) \qquad (4.1)$$

At first, a time step of 20 times the ADI time step was employed {GMRES (x/20)}, but it became apparent that this was too large to resolve the shock motion properly. A time step factor of 5 was found to be small enough to adequately resolve the physics of the problem, but the GMRES was not stable using less than 10 directions, which resulted in over a 100% increase in computer time. This illustrates the tradeoff

between having the large time step necessary for effective speedup with GMRES and the small enough time step to accurately model the physics of the problem.

The lift and pitching moment histories are plotted as a function of phase angle, ωt, and are compared with the Euler calculations by Steger [Ref. 20] in Figures 7 and 8.

Another case which was tested is a Navier-Stokes calculation for a NACA 0012 airfoil in the deep dynamic stall condition. The Mach number is 0.283, the Reynolds number is 3.45 million, and the reduced frequency is 0.151. The airfoil motion is defined by

$$\alpha = 15^{o} - 10^{o}\cos(\omega t) \qquad (4.2)$$

A time step factor of 20 was tried initially. To get a comparison with the original ADI code, 20 directions were run {GMRES (20/20)}. Note that this takes much longer than the original ADI code to run, due to the GMRES overhead. Figures 9, 10, and 11 compare the GMRES results with experimental results by McAlister et al (Ref. 21). While the GMRES (20/20) code does not get quantitatively good results, the result follows the experiments qualitatively. Thus, the GMRES (20/20) run was chosen as a benchmark to compare later runs to. Figure 12 gives the residual history of the GMRES (20/20) run.

The next series of runs were performed to see what sort of speedups were likely from GMRES. For this set, a time step of 20 times the ADI time step was used (i.e., GMRES (x/20)). The number of directions were set at 10 and 5. Results for lift, moment, and residual are shown in Figures 13, 14, and 15. These are plotted against time as it is easier to judge results in this way. The output shows that GMRES (10/20) is very nearly as good as (20/20), while accuracy begins to fall off in the (5/20) run. Timings for these runs are given in Table (IV.1).

	CPU seconds required for 1 cycle of pitch	% of ADI time
ADI	3958	100.0
GMRES (20/20)	5679	143.5
GMRES (5/20)	1971	49.8
GMRES (10/20)	3079	77.8

Table (IV.1) GMRES (x/20) timings

The next series of runs were done to see the effect of the time step on the GMRES solver. From the results of the last series, GMRES (x/2x) was chosen (number of directions equal to half of the time step factor). These results are shown in Figures 16, 17, 18, 19, 20, and 21. The results were split into two groups to keep the graphs legible. From these graphs, it can be seen that there is a tradeoff between accuracy of the GMRES iteration (which goes up with number of directions) and the time step necessary to resolve flow phenomena. From this series of runs, it appears that a time factor of 20 is the best choice in this case. Timings for these runs are given in Table (IV.2), though timings were not available for the GMRES (20/40) case. It is seen from this table how the GMRES overhead increases dramatically with the number of directions used.

	CPU seconds required for 1 cycle of pitch	% of ADI time
ADI	3958	100.0
GMRES (5/10)	3087	78.0
GMRES (10/20)	3079	77.8
GMRES (40/80)	3957	100.0

Table (IV.2) GMRES (x/2x) timings

Another experiment was tried to reduce the amount of memory required for the GMRES calculation. In this run, the GMRES iteration was carried out more than once per time step with less directions (e.g., two 5 direction iterations instead of one 10 direction iteration per time step). This is effectively doing a Newton iteration on top of the GMRES iteration. The advantage was that the memory necessary for the GMRES iteration was cut in half. The possible disadvantage was that the second set of GMRES directions were not necessarily orthogonal to the first set.

It was found that the 'restart' method worked better than the single step method for this case. The residual had much less 'noise' than before, and was lower. It was hypothesized that the restart method allows the GMRES solver to recover from a bad initial guess for the flow field at the new time level. Since the contribution of the higher directions are small compared to the initial directions, it is not too surprising that the residuals would be comparable. Figure 22 compares the residual histories of the two runs, while Fig. 23 shows the lift coefficient histories.

It was noticed that the number of directions needed for a given level of convergence was less in the portion of the cycle where the flow is attached. To take advantage of this, a switching mechanism based on residual was implemented in the restart solver. In this variant, the second GMRES iteration is not performed if the residual is below a user-specified tolerance. This resulted in a 30% speedup over the original restart code when a tolerance of 5×10^{-7} was input. Results of this run are given in Figures 24 and 25.

Timings for this series of runs are given in Table (IV.3).

	CPU seconds required for 1 cycle of pitch	% of ADI time
ADI	3958	100.0
GMRES (10/20)	3079	77.8
GMRES (5:5/20)	3110	78.6
GMRES (5dyn/20)	2644	66.8

Table (IV.3) Restart GMRES timings

Multigrid Analysis

At this point, a multigrid solver was introduced to try to reduce the number of GMRES directions necessary for convergence (and thus reduce the total memory required). In each iteration, the variables are transferred to a coarse grid and a GMRES iteration is performed there. It was hoped that this coarse grid calculation would aid in minimizing the low frequency error components, while the fine grid iterations reduced the high frequency error components. The multigrid solver used three 5 direction GMRES iterations per time step in a F-C-F pattern. In order to compare these with prior results, it was decided to use the same number of fine grid directions per iteration.

To validate the multigrid solver, the same steady runs were performed. The multigrid solver gave impressive speedups as compared to the fine-grid-only GMRES results. One noticeable difference was that the transonic steady case only took 5 directions to converge (down from 40 with only the fine grid). Residual histories for the steady runs are given in Fig. 26 and 27.

The multigrid solver was then run in unsteady mode on the dynamic stall test case. In Figures 28, 29, and 30, a (20/20) run is compared to a fine-grid-only (5:5/20) run

(two 5 direction iterations per time step) and a F-C-F (5:5/20) run (a 5 direction evaluation on the fine grid, then the coarse grid, then on the fine grid again). In effect, this is testing the effectiveness of the coarse grid evaluation. No appreciable gain due to multigrid was apparent except when the flow is attached and the flowfield is relatively smooth.

Table (IV.4) gives the timings for these runs.

	CPU seconds required for 1 cycle of pitch	% of ADI time
ADI	3958	100.0
GMRES (5:5/20)	3110	78.6
GMRES (5fcf/20)	3561	90.0

Parallel Steady Flow Analyses

First, the ADI code was implemented on a 32 processor Intel iPSC/860 MIMD parallel computer. Test runs for steady flow only were performed with from 4 to 32 processors.

Even though these runs were for steady flow cases, pseudo time-accuracy was a goal (i.e., the same answer being achieved on each iteration regardless of the number of processors employed). In order to accomplish this, the maximum change in Δq on the boundaries was computed, and ξ-sweep iterations were performed until the maximum change in Δq was less than 1% of its absolute value. This criteria proved to be adequate for a time-accurate computation; usually between 12 and 25 iterations were necessary for convergence, with the number of iterations increasing with the number of processors used. Using the values for Δq^* on the boundaries as the first guess for Δq proved to be the best

initial guess tested, and convergence was adequate, but an improved iteration procedure would speed the solver dramatically.

The problem chosen was the Navier-Stokes test problem; that of a NACA 0012 airfoil at a 5 degree angle of attack, with a freestream Mach number of 0.283 and Reynolds number of 3,450,000.

Figure 31 shows the residual histories of these runs for 4, 8, 16, and 32 processors. The speedup obtainable with larger numbers of processors can be seen, but it is also noted that the speedup factor is not ideal, as shown in Figure 32. The timings are given in Table (IV.5).

Figure 33 shows the moment coefficient histories as a function of the number of iterations required, and the pseudo time-accuracy of this code is illustrated; all results fall identically on the same line.

Number of Processors	CPU time required for 1000 iterations	CPU time for ideal speedup	CPU time required/ideal CPU time
4	6037	6037	1.0
8	3230	3018.5	1.070
16	1985	1509.25	1.315
32	1463	754.625	1.939

Table (IV.5): Parallel Iterative ADI Timings for 1000 Iterations

At this point, the GMRES scheme was added. Since time was not available to develop the parallel GMRES routine, these results are extremely preliminary.

Figure 34 shows the residual history of the 10 direction GMRES runs with 8 processors. In this figure, '8p10d/s' denotes the version with separate residuals for each processor, while '8p10d/g' represents the version with a global residual solver. The runs are compared to those of the standard ADI with 8 and 32 processors. Note that the 8 processor global GMRES solver is actually faster than the 32 processor standard code, while the separate GMRES code is slower than the 4 processor standard code. It is hypothesized that this is occurring because the separate GMRES code causes the processors to struggle against each other while trying to minimize the residual in their respective domains.

The next figures show the effect of the Block Cyclic Reduction tridiagonal matrix solver on the parallel code. The BCR routine required 2^n+1 computational points to run; therefore the number of i points was increased from 163 to 259 (since the i=1 and i=imax points are not computed by the tridiagonal solver).

The intial runs of the BCR algorithm were performed on the same case as the iterative code. Figure 35 shows the residual histories of these runs for 4, 8, 16, and 32 processors. The speedup obtainable with larger numbers of processors can be seen, but it is also noted that the speedup factor is not ideal, as shown in Figure 36. The timings are given in Table (IV.6).

Since the BCR routine is a direct solve, the answers are not compared because they are identical.

It is seen that the BCR routine is much faster than the iterative scheme, but the BCR routine does not scale as well with the number of processors used. This is because the number of messages increases with the number of processors; also, the number of idle processors increases.

Number of processors	CPU time/1000 iterations	Ideal CPU time	CPU time required/ideal CPU time
4	3460	3460	1.0
8	2068	1730	1.195
16	1396	865	1.613
32	1096	432.5	2.534

Table (IV.6): BCR ADI Timings for 1000 Iterations

The global GMRES code was also implemented on the BCR code, though memory limitations of the Intel iPSC/860 prevented more than 5 directions being used. It is seen that the code is converging even with such a low number of directions, but the convergence rate is understandably lower than if more directions were employed. Figure 37 compares the GMRES residual with that of the BCR code.

CHAPTER V

3-D RESULTS

In 3-D, both the ADI code and a new LU-SGS code was used, since a satisfactory level of convergence was not always obtainable by employing more GMRES directions.

Validation of GMRES Code

One steady-state case was run with GMRES to validate it against the original ADI code. It was felt that a single validation case was adequate.

The steady-state 3-D validation case was that of an F-5 wing at a zero degree angle of attack. The freestream Mach number was 0.9, with a Reynolds number of 11,000,000. The Baldwin-Lomax turbulence model was used.

First, the original ADI code was run with eigenvalue-scaled local time stepping. Then, a GMRES solver was implemented, using 20 directions.

The residual definition for the 3-D code was not immediately apparent, since the underlying ADI code is a hybrid formulation that sweeps in the spanwise direction. Usually, the original code is set to sweep from root to tip in one step; in the next step, the sweep is from tip to root.

The initial GMRES implementation had one sweep per function evaluation, with the sweep direction being changed between GMRES steps. This caused problems because the

problem to be minimized by the GMRES solver changed every step, and the GMRES solver oscillated from step to step.

Then a two-sweep function evaluation was tried. In this method, the GMRES residual is defined as the result of a root-to-tip sweep and a tip-to-root sweep (two ADI steps). This definition worked well, and was used for all steady-state problems.

Then, convergence problems with GMRES in 3-D were encountered. When coupled with the hybrid ADI solver, the GMRES (20) solver would converge for a short time, and then stall. The residual obtained was too high for a useful solution; in fact, spurious shocks were in the flow field.

Some directions given by the ADI solver were plotted, and they contained considerable high-frequency noise. It appeared that the higher directions contained only noise, and the weighting coefficients for these directions were very small.

Both the ADI and GMRES solvers have been applied to the three cases discussed below. A 121 x 19 x 41 grid was used for all viscous calculations, and a 121 x 19 x 21 grid was used for the inviscid case. All timings and memory requirements given are from the NASA-Langley Cray Y/MP using a single processor. All of the residuals shown are computed using the L_2 norm. All of the experimental results cited are from Tijdeman, et. al. (Ref. 20).

Steady Transonic Flow about an F-5 Wing

To validate the 3-D GMRES code, steady transonic flow about an F-5 wing at freestream Mach number of 0.9, Re = 11 x 10^6, and a = 0.0° was solved. The F-5 wing geometry contains large sweep, high taper, and drooped, sharp leading edges, and is a standard configuration recommended by AGARD for code validation. Initially, the noniterative ADI solver was run to get a baseline solution and estimates of CPU time.

Next, the GMRES routine was applied to the noniterative ADI solver, defining the function to be minimized as:

$$\mathbf{q}^{n+1} - \mathbf{q}^{n} = \mathbf{M}(\mathbf{q}^{n}) = 0 \qquad (5.1)$$

Initially, the same input settings (i.e., time step, dissipation coefficients, etc.) used by the noniterative ADI solver were employed in the iterative ADI function evaluator. This version was run with various numbers of GMRES directions. It quickly became apparent that the higher GMRES directions contained little useful information due to high-frequency noise, and this stalled the convergence before an adequate answer was obtained. In order to damp out the high-frequency noise, the implicit dissipation coefficients were increased in the ADI preconditioner, and a competitive convergence rate was achieved with the GMRES solver using 20 directions (referred to hereafter as GMRES (20)). The GMRES (20) solver had a slow initial convergence rate, but its asymptotic convergence rate at later iterations was comparable to the ADI scheme.

The ADI code required 2.77 megawords (mW) of memory to run, while the GMRES (20) code required 14.55 mW – a fivefold increase which is required for storing the 20 directions. Fig. 38 compares the L_2 norm of the residuals of the original ADI solver with the GMRES (20) results for various implicit dissipation ('ID' on the graph) coefficients. Fig. 39 shows the lift coefficient histories of these runs, and Fig. 40 compares the pressure coefficients at the 18.1% spanwise station given by the 20 direction GMRES solver with that of the original ADI solver and experiment.

From these calculations, it was concluded that the GMRES solver will give results identical to the ADI solver for steady state application. Since the goal of the present study

was to reduce the CPU time necessary for unsteady applications, this case was not pursued further to see if additional speedups using GMRES were possible.

LU-SGS Solver Applied to 3-D Steady Flows

At this point, the LU-SGS 3-D solver was applied to this steady flow problem. It should be noted that this implementation of the LU solver is far from optimal; a fully vectorized version would have required a total rewrite of the RHS (residual) calculation subroutines, which would have greatly increased the debugging time necessary. Instead, the LU solver was implemented only as a LHS replacement for the ADI solver, and evaluated as a preconditioner for the GMRES routine.

An initial run of the LU solver caused some concern because the L_2 norm of the residual did not drop monotonically as did the residual from the ADI solver. Instead, the residual would oscillate while generally decreasing. It was felt that this could cause problems such as the "stalling" phenomenon noticed with the ADI.

Due to time constraints, only the steady inviscid F5 wing case was run. In this case, the grid is identical, except that 21 normal points were used instead of 41. In order to minimize the stalling, the LU solver was run alone for 250 iterations from a cold start, and this solution was used to start the GMRES steady solver.

In Figure 41, the L_2 norm computed by the original LU solver is compared to that of the GMRES (20) and GMRES (5) schemes. All of these runs use a β parameter of 1.0 and an explicit dissipation parameter of 0.01. It is seen that the 5 direction solver stalls within 4 steps, and the residual is never reduced after that. The 20 direction solver however, does succeed in reducing the residual slightly faster than the original LU solver.

To determine if more speedup is possible with the GMRES solver, the β parameter was varied between 0.65 and 1.0. It was seen that a higher value generally retarded the convergence, and a lower value increased the convergence rate. It was found that even

though the original code was unstable with $\beta = 0.65$, the GMRES procedure stabilized the code; however, the convergence rate was reduced with this low value of β. It was found that a β value around 0.70 gave the best GMRES convergence rate, and this is illustrated in Figure 42.

For unsteady flow problems, an earlier version of the LU solver was evaluated and found to show no advantages over the existing 3-D ADI code. Therefore, research on the LU solver for unsteady flows was not pursued.

Unsteady Viscous Flow about an F-5 Wing with an Oscillating Flap

The second case investigated is the unsteady flow over an F-5 wing with a harmonically oscillating trailing edge control surface, hinged at x/c = 0.82. The trailing edge oscillates at a frequency of 20 Hz, the $M_\infty = 0.90$, $Re = 11.0 \times 10^6$, $a_{wing} = 0.0°$, and $a_{flap} = \pm 0.5°$.

The unsteady pressure coefficients from the ADI calculation compared with the experimental data for this case are shown in Fig. 43 and 44. In the comparison, the real and imaginary components of the pressure coefficients are defined as:

$$(C_p)_{imaginary} = \frac{(C_p)_{\omega t = \pi} - (C_p)_{\omega t = 0}}{2\Delta\alpha}$$

$$(C_p)_{real} = \frac{(C_p)_{\omega t = \frac{3\pi}{2}} - (C_p)_{\omega t = \frac{\pi}{2}}}{2\Delta\alpha} \qquad (5.2)$$

The data presented in Fig. 43 and 44 are for the initial 3/4 cycle of oscillation, at the 18.1% span station. Our studies with the noniterative ADI solver indicate that even better

correlation with the experimental data can be achieved if the solution is allowed to march more than one cycle or until no discrepancies are found between successive cycles of oscillation.

The preconditioner for the GMRES calculation was the iterative ADI solver described in Chapter 3. Within each time level, local time steps are used for the sub-iterations. That is, Eq. (3.6) is replaced by:

$$\left\{I + \Delta t_{i,j,k}\partial_\xi A^{n+1,k} - \left(\varepsilon_I\right)_\xi\right\}\left\{I + \Delta t_{i,j,k}\partial_\zeta C^{n+1,k} - \left(\varepsilon_I\right)_\zeta\right\}\left\{\Delta q\right\}$$
$$= -\Delta t_{i,j,k}\left(\frac{q^{n+1,k} - q^{n}}{\Delta t}\right) + \Delta t_{i,j,k}\left[\partial_\xi\left(R - E\right) + \partial_\eta\left(S - F\right) + \partial_\zeta\left(T - E\right)\right]^{n+1,k} \qquad (5.3)$$

where $\Delta t_{i,j,k}$ is the local time step, which is a function of the grid and local flow conditions.

For initial comparisons, a five direction GMRES run was made at five times the ADI time step (GMRES (5/5), where the first number designates the number of directions, and the second number is the time step factor), and the GMRES solution followed the ADI solution exactly.

After the initial validation, the GMRES time step factor was increased to numerically determined the largest time step that can be used without large loss of accuracy due to temporal discretization errors. To carry out this task, GMRES (5/10) and (5/20) runs were performed. Both the (5/10) and (5/20) runs gave good results while providing significant speedups, but the (5/20) results showed some degradation in solution accuracy. The CPU time and memory required for the ADI and three GMRES runs are shown in Table (V.1).

	Memory (mW)	CPU time (sec)	CPU (% of ADI)
ADI	3.66	5533	100
GMRES (5/10)	7.72	3952	71
GMRES (5/20)	7.72	2002	36

Table (V.1): Unsteady Transonic Viscous Flow Computer Requirements

Time histories of the mid half-span moment coefficient for two GMRES runs are compared in Fig. 45. It is seen that the results are identical to that of the ADI solver. Figure 46 shows the residual histories of the GMRES runs, and Figures 47 and 48 compare the real and imaginary components of the pressure coefficients with both experiment and the ADI solver.

It should be remembered that the imaginary component of the pressure coefficient is measured at the times when the flap is moving the fastest. Therefore, the imaginary component of the pressure coefficient is the best measure of the time accuracy of the code. Conversely, the real component of the pressure coefficient is measured when the flap is moving the slowest, and is a much looser measure of time accuracy.

Unsteady Flow about an F-5 Wing in Modal Vibration

The third case investigated is the unsteady inviscid flow about an F-5 wing undergoing modal vibration. In this case, $M\infty = 0.90$, $a_{wing} = 0.0°$, $a_{oscillation} = \pm 0.5°$, with an oscillation frequency of 40Hz (reduced frequency of 0.275). The amplitude of the wing surface deformation is defined by:

$$w(x,y) = -0.329 + 0.977x - 0.088y + 0.244xy - 0.077y^2 - 0.091xy^2 \qquad (5.4)$$

Eq. (5.4) gives a pure angular displacement with the nondimensionalization performed such that the tangent of the angle of oscillation at experimental span station 2 is equal to one. The pressure coefficient may be separated into real and imaginary components by using Eq. (5.2).

The results of the original ADI code are compared to experiment in Figures 49 and 50. The ADI code required 1.5 mW of memory to run, and took 698 CPU seconds to complete 3/4 of a cycle of oscillation.

Since this simulation requires very little CPU time, it was used to more thoroughly determine the effects of both the time step and the error at each step on the solution accuracy.

Again, GMRES (5/5) was used as an initial run, and the results were identical to the original ADI code. To limit the GMRES memory requirements, only 5 directions were employed. The 5 direction GMRES code required 4.1 mW of memory to run (2.73 times larger than the original ADI code).

Effects of Time Step on Solution with Five Directions

At this point, the time step was increased to determine the maximum time steps possible with 5 GMRES directions. Time steps that were 10, 20, and 40 times larger than those used by the ADI scheme were tried. Since shock speed is sensitive to time step size, and critically affects pitching moment, the mid-halfspan moment coefficient histories were used to study the effects on solution accuracy, as shown in Fig. 51. The residual histories of the GMRES runs are shown in Fig. 52. The real and imaginary components of the

pressure coefficient at the 18.1% spanwise station are shown in Fig. 53, 54, 55, and 56. These graphs are split for legibility.

These figures show that the solution begins to degrade slightly at 20 times the ADI time step, and the moment coefficient, influenced by shock speed, is very different at 40 times the time step. To put this into perspective, the nondimensionalized ADI time step is

$$\Delta\tau = \frac{(a\Delta t)}{c} = 0.1 \tag{5.5}$$

One complete cycle of harmonic oscillation requires 1270 time steps, which is 0.283 degrees of harmonic oscillation per time step. In this manner, it is seen that a GMRES (5/40) computation takes only 32 time steps per cycle, which is 11.33 degrees of harmonic oscillation per step. With such a large time step, an error in shock speed is not entirely unexpected.

	Memory (mW)	CPU time (sec)	CPU (% of ADI)
ADI	1.4	698	100
GMRES (5/10)	4.1	513	74
GMRES (5/20)	4.1	265	38
GMRES (5/40)	4.1	131	19

Table (V.2): Unsteady Inviscid Transonic Flow Computer Requirements

The CPU time and memory required for the ADI and three GMRES runs are shown in Table (V.2).

Effect of Residual Magnitude on Solution Accuracy (Time Step Fixed)

The next area of investigation is to determine if a larger time step may be employed if the error is reduced more at each time step. Using the GMRES solver, there are two ways to accomplish this: either use more directions in each iteration, or perform more than one GMRES iteration at each time step ('restart' GMRES). Restart GMRES is discussed in detail in Chapter 3.

	Memory (mW)	CPU time (sec)	CPU (% of ADI)
ADI	1.4	698	100
GMRES (5/40) -1	4.1	131	19
GMRES (5/40) -2	4.1	262	38
GMRES (5/40) -3	4.1	390	56
GMRES (5/40) -4	4.1	525	75

Table (V.3): CPU time and memory Usage for ADI and GMRES Calculations for Flow about an F5 Wing in Modal Vibration

Restart GMRES was chosen in order to keep the memory requirements constant. The restart code was employed on the GMRES (5/40) run, and up to four 5-direction GMRES iterations were used per time step (GMRES (5/40) - 1, (5/40) - 2, etc.). As more iterations were used, and the error residual decreased, the answer approached the ADI solution, but smeared out the pressure peaks. Figures 57 and 58 compare the imaginary

components of the pressure coefficients computed by the restart GMRES (5/40) - x code at the 18.1% span station. Figure 59 shows the residual histories of the GMRES (5/40) - x runs.

The CPU time and memory required for the ADI and three GMRES runs are shown in Table (V.3).

Effect of Time Step on Solution Accuracy (Magnitude Fixed)

The next part of this investigation was to compare various GMRES runs which use different time steps but result in the same error magnitude. This would isolate the effect of the time step on the unsteady solution. To illustrate the results, a GMRES (5/20) - 2 run is compared to a GMRES (5/40) - 4 code that achieved almost identical error residuals. Fig. 60 shows the imaginary component of the pressure coefficient at the 18.1% span station. It is shown that even at this very large time step, the GMRES code still resolves the shock location well, but the shock is somewhat smeared as the time step is increased.

The CPU time and memory required for the ADI and GMRES runs are shown in Table (V.4).

	Memory (mW)	CPU time (sec)	CPU (% of ADI)
ADI	1.4	698	100
GMRES (5/20) -2	4.1	536	77
GMRES (5/40) -4	4.1	525	75

Table (V.4): CPU time and memory usage for ADI and GMRES calculations for flow about an F5 wing in modal vibration

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

The GMRES algorithm was implemented on unsteady compressible viscous flow solvers in both two and three dimensions, and reduced the CPU time necessary for these computations by nearly 60% in most cases. The drawback to the GMRES procedure is the increased storage required by the search directions.

It was shown that a Newton/ADI procedure was an effective preconditioner for GMRES, but a simpler preconditioner such as an LU-SGS scheme may prove to be more efficient. A 3-D LU-SGS solver was implemented, but was not vectorized to take full advantage of the LU-SGS algorithm. With a fully vectorized code, however, the LU-SGS scheme may well be competitive.

In the 2-D code, restart GMRES was investigated, and found to be an effective way to cut the memory requirements of the GMRES method while not requiring much more CPU time. Also, multigrid methods were implemented, and while they greatly improved steady-state convergence, in unsteady applications it was found that the greater accuracy did not warrant the extra CPU time required.

The restart GMRES also was effective in 3-D, and the memory requirements were only about twice that of the original ADI code. The CPU time required by the GMRES 3-D code was as low as 36% of that of the ADI code, while still retaining good accuracy.

A parallel version of GMRES was implemented on the NASA-Langley Intel iPSC/860, and preliminary results were obtained. The global GMRES did very well, but

the separate GMRES did not do well at all. It was also found that a Block Cyclic Reduction routine sped up the function evaluation procedure, but that the memory required by the BCR routine reduced the number of GMRES directions that could be used.

In the future, the GMRES research on sequential computers should concentrate on testing various preconditioners to establish the most effective ones. The GMRES research on parallel computers should not only investigate alternate preconditioners that are more parallelizable, but also investigate more efficient ways of inverting the tridiagonal matrices encountered in ADI solution procedures.

Finally, after more 2-D experience is gained and a machine with more memory is available, the parallel code should be rewritten in 3-D. With this version of the code, extension to 3-D is straightforward.

APPENDIX A

TRANSFORMATION TO CURVILINEAR COORDINATES

This appendix details the transformation of the governing equations from Cartesian to generalized curvilinear coordinates. For simplicity and to save space, only the 2-D equations are considered; extension to 3-D is straightforward.

The Navier-Stokes equations written in Cartesian coordinates are:

$$\partial_t q + \partial_x E + \partial_z G = \frac{1}{Re}\left\langle \partial_x R + \partial_z T \right\rangle . \qquad (A.1)$$

The transformed coordinates are:

$$\begin{aligned} \xi &= \xi\langle x,z,t\rangle \\ \zeta &= \zeta\langle x,z,t\rangle \\ \tau &= t \end{aligned} \qquad (A.2)$$

From the chain rule:

$$\begin{aligned} \partial_t &= \partial_\tau + \xi_t \partial_\xi + \zeta_t \partial_\zeta \\ \partial_x &= \xi_x \partial_\xi + \zeta_x \partial_\zeta \\ \partial_z &= \xi_z \partial_\xi + \zeta_z \partial_\zeta \end{aligned} \qquad (A.3)$$

Applying Eq. (A.3) to the governing equations gives:

$$\begin{aligned} &\partial_\tau q + \xi_t \partial_\xi q + \zeta_t \partial_\zeta q \\ &+\xi_x \partial_\xi E + \zeta_x \partial_\zeta E + \xi_z \partial_\xi G + \xi_x \partial_\xi G = \\ &\frac{1}{Re}\left\langle \xi_x \partial_\xi R + \xi_x \partial_\xi R + \xi_x \partial_\xi T + \zeta_z \partial_\zeta T \right\rangle . \end{aligned} \qquad \text{(A.4)}$$

At this point, it is noticed that the numerical evaluation of the transformed derivatives (x_τ , x_ξ ,x_ζ , etc.) will have the same problems with computational accuracy that the original equations did (i.e., they must be computed on a stretched, non-Cartesian grid). To avoid this problem, the derivatives of the transformation variables (which are evaluated in the physical plane) are rewritten in terms of the derivatives of the Cartesian grid variables (which are evaluated in the computational plane, and thus are more accurate). This transformation is performed as follows:

Eq. (A.3) is written in matrix form as:

$$\begin{Bmatrix} \partial_t \\ \partial_x \\ \partial_z \end{Bmatrix} = \begin{bmatrix} 1 & \xi_t & \zeta_t \\ 0 & \xi_x & \zeta_x \\ 0 & \xi_z & \zeta_z \end{bmatrix} \begin{Bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\zeta \end{Bmatrix} . \qquad \text{(A.5)}$$

Or, by using the chain rule to find the curvilinear derivatives in terms of the Cartesian derivatives, one obtains:

$$\begin{Bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\zeta \end{Bmatrix} = \begin{bmatrix} 1 & x_\tau & z_\tau \\ 0 & x_\xi & x_\zeta \\ 0 & z_\xi & z_\zeta \end{bmatrix} \begin{Bmatrix} \partial_t \\ \partial_x \\ \partial_z \end{Bmatrix}. \tag{A.6}$$

Solving Eq. (A.6) for the Cartesian derivatives gives:

$$\begin{Bmatrix} \partial_t \\ \partial_x \\ \partial_z \end{Bmatrix} = J \begin{bmatrix} (x_\xi z_\zeta - z_\xi x_\zeta) & (-x_\tau z_\zeta - z_\tau x_\xi) & (-x_\tau z_\xi - z_\tau x_\zeta) \\ 0 & z_\zeta & -z_\xi \\ 0 & -x_\zeta & x_\xi \end{bmatrix} \begin{Bmatrix} \partial_\tau \\ \partial_\xi \\ \partial_\zeta \end{Bmatrix} \tag{A.7}$$

where J is the Jacobian of transformation and is defined:

$$J = \frac{1}{x_\xi z_\zeta - z_\xi x_\zeta} \tag{A.8}$$

From comparing Eq. (A.5) and (A.7), the following definitions are found:

$$\begin{aligned} \xi_x &= J z_\zeta \\ \xi_z &= -J x_\zeta \\ \xi_t &= -x_\tau \xi_x - z_\tau \xi_z \\ \zeta_x &= -J y_\xi \\ \zeta_z &= J x_\xi \\ \zeta_t &= -x_\tau \zeta_x - z_\tau \zeta_z \end{aligned} \tag{A.9}$$

At this point, it should be noted that Eq. (A.4), which is the transformed governing equation, is not written in conservation form (there are non-constant terms outside the flux and viscous derivatives). In order to express the transformed governing equation in conservation form, the following derivation is performed:

First, Eq. (A.4) is multiplied by the reciprocal of the Jacobian (1/J). Then the chain rule is applied to the resulting terms. For example, the second term term becomes:

$$\frac{1}{J}\xi_t \partial_\xi q = \partial_\xi\left(\frac{1}{J}\xi_t q\right) - q\,\partial_\xi\left(\frac{1}{J}\xi_t\right) \tag{A.10}$$

After the chain rule is applied, Eq. (A.4) becomes:

$$\begin{aligned}
&\partial_\tau\left(\frac{q}{J}\right) + \partial_\xi\left(\frac{\xi_t q}{J}\right) + \partial_\zeta\left(\frac{\zeta_t q}{J}\right) - q(M_1) \\
&+ \partial_\xi\left(\frac{\xi_x E}{J}\right) + \partial_\zeta\left(\frac{\zeta_x E}{J}\right) - E(M_2) \\
&+ \partial_\xi\left(\frac{\xi_z G}{J}\right) + \partial_\zeta\left(\frac{\zeta_z G}{J}\right) - G(M_3) \\
&= \frac{1}{Re}\begin{Bmatrix} \partial_\xi\left(\frac{\xi_x R}{J}\right) + \partial_\zeta\left(\frac{\zeta_x R}{J}\right) - R(M_2) \\ + \partial_\xi\left(\frac{\xi_z T}{J}\right) + \partial_\zeta\left(\frac{\zeta_z T}{J}\right) - T(M_3) \end{Bmatrix}
\end{aligned} \tag{A.11}$$

where

$$M_1 = \partial_\tau\left(\frac{1}{J}\right) + \partial_\xi\left(\frac{\xi_t}{J}\right) + \partial_\zeta\left(\frac{\zeta_t}{J}\right)$$
$$M_2 = \partial_\xi\left(\frac{\xi_x}{J}\right) + \partial_\zeta\left(\frac{\zeta_x}{J}\right)$$
$$M_3 = \partial_\xi\left(\frac{\xi_z}{J}\right) + \partial_\zeta\left(\frac{\zeta_z}{J}\right) \qquad \text{(A.12)}$$

These expressions for M_1, M_2, and M_3 are equal to zero. This is shown by using Eq. (A.9) to rewrite Eq. (A.12) as:

$$M_1 = \partial_\tau\left(x_\xi z_\zeta - z_\xi x_\zeta\right) + \partial_\xi\left(-x_\tau z_\zeta + z_\tau x_\zeta\right) + \partial_\zeta\left(x_\tau z_\xi - z_\tau x_\xi\right) = 0$$
$$M_2 = \partial_\xi\left(z_\zeta\right) + \partial_\zeta\left(-z_\xi\right) = z_{\zeta\xi} - z_{\xi\zeta} = 0$$
$$M_3 = \partial_\xi\left(-x_\zeta\right) + \partial_\zeta\left(x_\xi\right) = x_{\xi\zeta} - x_{\zeta\xi} = 0 \qquad \text{(A.13)}$$

Thus, these terms are dropped. Rewriting Eq. (A.11) without these terms, one obtains:

$$\partial_\tau\left(\frac{q}{J}\right) + \partial_\xi\left(\frac{\xi_t q}{J}\right) + \partial_\zeta\left(\frac{\zeta_t q}{J}\right) + \partial_\xi\left(\frac{\xi_x E}{J}\right) + \partial_\zeta\left(\frac{\zeta_x E}{J}\right) + \partial_\xi\left(\frac{\xi_z G}{J}\right) + \partial_\zeta\left(\frac{\zeta_z G}{J}\right)$$

$$= \frac{1}{Re}\begin{Bmatrix} \partial_\xi\left(\frac{\xi_x R}{J}\right) + \partial_\zeta\left(\frac{\zeta_x R}{J}\right) \\ + \partial_\xi\left(\frac{\xi_z T}{J}\right) + \partial_\zeta\left(\frac{\zeta_z T}{J}\right) \end{Bmatrix} \qquad \text{(A.14)}$$

These terms may be regrouped to give

$$\partial_t q + \partial_\xi E + \partial_\zeta G = \frac{1}{Re}\left\langle \partial_\xi R + \partial_\zeta T \right\rangle \qquad \text{(A.15)}$$

where:

$$q = \frac{1}{J}\begin{pmatrix} \rho \\ \rho u \\ \rho w \\ e \end{pmatrix};\ E = \frac{1}{J}\begin{pmatrix} \rho u \\ \rho u U + \xi_x p \\ \rho w U + \xi_z p \\ U\langle e+p\rangle - \xi_t p \end{pmatrix};\ G = \frac{1}{J}\begin{pmatrix} \rho w \\ \rho u W + \zeta_x p \\ \rho w W + \zeta_z p \\ W\langle e+p\rangle - \zeta_t p \end{pmatrix}$$

$$R = \frac{1}{J}\begin{pmatrix} 0 \\ \xi_x \tau_{xx} + \xi_z \tau_{xz} \\ \xi_x \tau_{xz} + \xi_z \tau_{zz} \\ \xi_x R_4 + \xi_z T_4 \end{pmatrix};\ T = \frac{1}{J}\begin{pmatrix} 0 \\ \zeta_x \tau_{xx} + \zeta_z \tau_{xz} \\ \zeta_x \tau_{xz} + \zeta_z \tau_{zz} \\ \zeta_x R_4 + \zeta_z T_4 \end{pmatrix} \qquad \text{(A.17)}$$

with the contravariant velocities U and V given as:

$$U = \xi_t + \xi_x u + \xi_z w$$
$$W = \zeta_t + \zeta_x u + \zeta_z w \qquad \text{(A.18)}$$

The transformed viscous terms are:

$$\tau_{xx} = \mu\left[\frac{4}{3}\left(\xi_x u_\xi + \zeta_x u_\zeta\right) - \frac{2}{3}\left(\xi_z w_\xi + \zeta_z w_\zeta\right)\right]$$
$$\tau_{xz} = \mu\left[\left(\xi_z u_\xi + \zeta_z u_\zeta\right) + \left(\xi_x w_\xi + \zeta_x w_\zeta\right)\right]$$
$$\tau_{zz} = \mu\left[\frac{4}{3}\left(\xi_z w_\xi + \zeta_z w_\zeta\right) - \frac{2}{3}\left(\xi_x u_\xi + \zeta_x u_\zeta\right)\right]$$
$$R_4 = u\tau_{xx} + w\tau_{xz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_x \partial_\xi a^2 + \zeta_x \partial_\zeta a^2\right)$$
$$T_4 = u\tau_{xz} + w\tau_{zz} + \frac{\mu}{\Pr(\gamma-1)}\left(\xi_z \partial_\xi a^2 + \zeta_z \partial_\zeta a^2\right) \qquad \text{(A.19)}$$

APPENDIX B

DISCRETIZATION IN TIME AND SPACE

The 2-D Reynolds averaged Navier-Stokes equations written in curvilinear form are given as:

$$\partial_\tau q + \partial_\xi E + \partial_\zeta G = \frac{1}{Re}\left\langle \partial_\xi R + \partial_\zeta T \right\rangle \qquad \text{(B.1)}$$

This equation is discretized by using the Euler implicit scheme, which is first order accurate in time and second order accurate in space. The time derivative is approximated by a first order forward difference. Using Taylor series expansion:

$$\begin{aligned} \partial_\tau q^n = \vec{\delta}_\tau q^n + O(\Delta\tau^2) &= \frac{(q^{n+1,k+1} - q^n)}{\Delta\tau} + O(\Delta\tau^2) \\ &= \frac{(q^{n+1,k+1} - q^{n+1,k} - q^n + q^{n+1,k})}{\Delta\tau} + O(\Delta\tau^2) \\ &= \frac{\Delta q^{n+1,k}}{\Delta\tau} + \frac{(q^{n+1,k} - q^n)}{\Delta\tau} + O(\Delta\tau^2) \end{aligned} \qquad \text{(B.2)}$$

where $O(\Delta\tau^2)$ indicates that this expression is first order accurate in time (second order terms are truncated). In this expression, $\Delta q^{n+1,k}$ is the change in the flow properties between the 'k' and 'k+1' iteration levels, and 'n' is the old time level (at which the flow properties are known), while 'n+1' is the new time level where the iteration is taking place.

Using Eq.(A.2), the unknown time level 'n+1' can be computed using the flow properties at the known 'n' time level.

The spatial derivative terms are approximated with second order accurate central difference operators. For example,

$$\partial_\xi \mathbf{E} = \delta_\xi \mathbf{E} + O\left(\Delta\xi^3\right) = \frac{\mathbf{E}_{i+1,j} - \mathbf{E}_{i-1,j}}{2\Delta\xi} + O\left(\Delta\xi^3\right) \qquad \text{(B.3)}$$

In the computational plane, $\Delta\xi$ is taken as one and i,j is a grid node point.

With this discretization method, the computational stencil for the convective terms **E** and **G** depend only on the values of the variables at the grid points adjacent to the node being computed. In order to have the same stencil with the viscous terms **R** and **T**, the half points between the nodes are used.

The viscous terms ($\delta_\xi \mathbf{R}$ and $\delta_\zeta \mathbf{T}$) in the governing equations all have the form $\delta_\xi(c\delta_\xi u)$, where c consists of the metrics of transformation and the viscosity. These terms are discretized using the values of the derivatives at the half points surrounding the node being computed:

$$\delta_\xi\left(c\delta_\xi u\right) \cong \partial_\xi\left(c\partial_\xi u\right)_{i,j}$$

$$= \frac{\left[\left(c\partial_\xi u\right)_{i+\frac{1}{2},j} - \left(c\partial_\xi u\right)_{i-\frac{1}{2},j}\right]}{\Delta\xi}$$

$$= \frac{\left[c_{i+\frac{1}{2},j}\dfrac{\left(u_{i+1,j} - u_{i,j}\right)}{\Delta\xi} - c_{i-\frac{1}{2},j}\dfrac{\left(u_{i,j} - u_{i-1,j}\right)}{\Delta\xi}\right]}{\Delta\xi}$$

$$= \frac{\left[\frac{(c_{i+1,j} + c_{i,j})(u_{i+1,j} - u_{i,j})}{2} - \frac{(c_{i,j} + c_{i-1,j})(u_{i,j} - u_{i-1,j})}{2}\right]}{\Delta\xi}$$

$$= \frac{1}{2\Delta\xi}\left[(c_{i+1,j} + c_{i,j})(u_{i+1,j} - u_{i,j}) - (c_{i,j} + c_{i-1,j})(u_{i,j} - u_{i-1,j})\right] \quad \text{(B.4)}$$

This discretization gives a compact three point stencil. As before, the grid spacing on the computational plane is taken to be unity.

Substituting into Eq. (B.1), the discretized equation becomes:

$$\delta_\tau q^n + \Delta\tau\left(\delta_\xi E^{n+1} + \delta_\zeta G^{n+1}\right) = \frac{\Delta\tau}{Re}\left(\delta_\xi R^{n+1} + \delta_\zeta T^{n+1}\right) + O\left(\Delta\tau^2, \Delta\xi^3, \Delta\zeta^3\right) \quad \text{(B.5)}$$

Similarly, the 3-D equation becomes:

$$\delta_\tau q^n + \Delta\tau\left(\delta_\xi E^{n+1} + \delta_\eta F^{n+1} + \delta_\zeta G^{n+1}\right) =$$
$$\frac{\Delta\tau}{Re}\left(\delta_\xi R^{n+1} + \delta_\eta S^{n+1} + \delta_\zeta T^{n+1}\right) + O\left(\Delta\tau^2, \Delta\xi^3, \Delta\eta^3, \Delta\zeta^3\right) \quad \text{(B.6)}$$

APPENDIX C

LINEARIZATION OF THE DISCRETIZED EQUATIONS

The discretized equation from before is:

$$\delta_\tau q^n + \Delta\tau\left(\delta_\xi E^{n+1} + \delta_\zeta G^{n+1}\right) = \frac{\Delta\tau}{Re}\left(\delta_\xi R^{n+1} + \delta_\zeta T^{n+1}\right) + O\left(\Delta\tau^2, \Delta\xi^3, \Delta\zeta^3\right) \qquad \text{(C.1)}$$

Given the flow variables at the 'n' time level, equation set (C.1) can now be solved to obtain the flow variables at the 'n+1' time level. Unfortunately, this set of algebraic equations are coupled and highly nonlinear, making them very difficult to solve. To make these equations easier to solve, the convection terms **E** and **G** are linearized about time level 'n+1' and iteration level 'k' by means of Taylor series:

$$\mathbf{E}^{n+1,k+1} = \mathbf{E}^{n+1,k} + \left(\frac{\partial \mathbf{E}}{\partial \mathbf{q}}\right)^{n+1,k} \Delta\mathbf{q}^{n+1,k} + O\left(\Delta\tau^2\right)$$

$$= \mathbf{E}^{n+1,k} + \mathbf{A}^{n+1,k}\Delta\mathbf{q}^{n+1,k} + O\left(\Delta\tau^2\right) \qquad \text{(C.2)}$$

$$\mathbf{G}^{n+1,k+1} = \mathbf{G}^{n+1,k} + \left(\frac{\partial \mathbf{G}}{\partial \mathbf{q}}\right)^{n+1,k} \Delta\mathbf{q}^{n+1,k} + O\left(\Delta\tau^2\right)$$

$$= \mathbf{G}^{n+1,k} + \mathbf{C}^{n+1,k}\Delta\mathbf{q}^{n+1,k} + O\left(\Delta\tau^2\right) \qquad \text{(C.3)}$$

where **A** and **C** are the Jacobian matrices of the flux vectors **E** and **G**. These matrices are defined:

$$\mathbf{A} = \frac{\partial \mathbf{E}}{\partial \mathbf{q}} = \xi_t \mathbf{I} + \xi_x \frac{\partial \mathbf{E}}{\partial \mathbf{q}} + \xi_z \frac{\partial \mathbf{G}}{\partial \mathbf{q}}$$
$$\mathbf{C} = \frac{\partial \mathbf{G}}{\partial \mathbf{q}} = \zeta_t \mathbf{I} + \zeta_x \frac{\partial \mathbf{E}}{\partial \mathbf{q}} + \zeta_z \frac{\partial \mathbf{G}}{\partial \mathbf{q}} \qquad \text{(C.4)}$$

where I is the 4x4 identity matrix; $\frac{\partial E}{\partial q}$ and $\frac{\partial G}{\partial q}$ are the flux Jacobian matrices of E and G with respect to q. After evaluating these and substituting into Eq. (C.4), **A** and **C** become:

$$\text{A or C} = \begin{bmatrix} k_t & k_x & k_z & 0 \\ k_x\phi^2 - u\theta & k_t + \theta - k_x\gamma_1 u & k_z u - k_x\gamma_1 w & k_x\gamma_1 \\ k_z\phi^2 - w\theta & k_x w - k_z\gamma_1 u & k_t + \theta - k_z\gamma_2 w & k_z\gamma_1 \\ -\theta\left(b_1 - \phi^2\right) & k_x b_1 - \gamma_1 u\theta & k_z b_1 - \gamma_1 v\theta & k_t + \gamma\theta \end{bmatrix} \qquad \text{(C.5)}$$

where

$$b_1 = \gamma\left(\frac{e}{\rho}\right) - \phi^2$$
$$\gamma_1 = \gamma - 1$$
$$\gamma_2 = \gamma - 2$$
$$\phi^2 = \frac{(\gamma - 1)\left(u^2 + w^2\right)}{2}$$

$$\theta = k_x u + k_z w$$
$$k = \begin{Bmatrix} \xi \text{ for } A \\ \zeta \text{ for } C \end{Bmatrix} \qquad \text{(C.6)}$$

Since these expansions are also first order accurate in time, this linearization will not affect the time accuracy of the solution.

The viscous terms are lagged to the 'k' iteration level, as their magnitude is small at high Reynolds numbers.

When Eqs. (C.2) and (C.3) are substituted into (C.1), the linearized equations are written as:

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k} + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} =$$
$$-\Delta\tau\left(\frac{q^{n+1,k} - q^n}{\Delta\tau}\right) - \Delta\tau\left\{\delta_\xi E^{n+1,k} + \delta_\zeta G^{n+1,k}\right\}$$
$$+\frac{\Delta\tau}{Re}\left\{\delta_\xi R^{n+1,k} + \delta_\zeta T^{n+1,k}\right\} \qquad \text{(C.7)}$$

This equation set is first order accurate in time and second order accurate in space.

In 3-D, the derivation is similar. Since 3-D ADI is at best neutrally stable, an explicit sweeping procedure is performed in the spanwise direction (η), using updated flow variables as they become available. This allows a 2D ADI problem to be solved at each spanwise plane. Thus, Eq. (C.7) becomes:

$$\left\{I + \Delta\tau\delta_\xi \mathbf{A}^{n+1,k} + \Delta\tau\delta_\zeta \mathbf{C}^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} =$$
$$-\Delta\tau\left(\frac{q^{n+1,k} - q^{n}}{\Delta\tau}\right) - \Delta\tau\left(\delta_\xi \mathbf{E}^{n+1,k} + \delta_\eta \mathbf{F}^{(n,n+1),k} + \delta_\zeta \mathbf{G}^{n+1,k}\right)$$
$$+\frac{\Delta\tau}{Re}\left(\delta_\xi \mathbf{R}^{n+1,k} + \delta_\eta \mathbf{S}^{n+1,k} + \delta_\zeta \mathbf{T}^{n+1,k}\right) \qquad \text{(C.8)}$$

where the superscript (n,n+1) is used to represent the explicit spanwise sweep.

In 3-D, the definitions of **A** and **C** change:

$$\mathbf{A} = \frac{\partial \mathbf{E}}{\partial q} = \xi_t \mathbf{I} + \xi_x \frac{\partial \mathbf{E}}{\partial q} + \xi_y \frac{\partial \mathbf{F}}{\partial q} + \xi_z \frac{\partial \mathbf{G}}{\partial q}$$
$$\mathbf{C} = \frac{\partial \mathbf{G}}{\partial q} = \zeta_t \mathbf{I} + \zeta_x \frac{\partial \mathbf{E}}{\partial q} + \zeta_y \frac{\partial \mathbf{F}}{\partial q} + \zeta_z \frac{\partial \mathbf{G}}{\partial q} \qquad \text{(C.9)}$$

and **A** and **C** become:

$$\mathbf{A} \text{ or } \mathbf{C} = \begin{bmatrix} k_t & k_x & k_y & k_z & 0 \\ k_x\phi^2 - u\theta & k_t + \theta - k_x y_2 \mu & k_y\mu - k_y y_1 v & k_z\mu - k_z y_1 w & k_x y_1 \\ k_y\phi^2 - v\theta & k_x v - k_y y_1 \mu & k_t + \theta - k_y y_2 v & k_z v - k_y y_1 w & k_y y_1 \\ k_z\phi^2 - w\theta & k_x w - k_z y_1 \mu & k_y w - k_z y_1 v & k_t + \theta - k_z y_2 w & k_z y_1 \\ -\theta\left(b_1 - \phi^2\right) & k_x b_1 - \gamma_1 u\theta & k_y b_1 - \gamma_1 \mu\theta & k_z b_1 - \gamma_1 w\theta & k_t + \gamma\theta \end{bmatrix} \qquad \text{(C.10)}$$

where:

$$b_1 = \gamma\left(\frac{e}{\rho}\right) - \phi^2$$

$$\gamma_1 = \gamma - 1$$

$$\gamma_2 = \gamma - 2$$

$$\phi^2 = \frac{(\gamma - 1)\left(u^2 + v^2 + w^2\right)}{2}$$

$$\theta = k_x u + k_y v + k_z w$$

$$k = \begin{Bmatrix} \xi \text{ for } A \\ \zeta \text{ for } C \end{Bmatrix} \qquad \text{(C.11)}$$

APPENDIX D

SOLUTION PROCEDURE FOR APPROXIMATE FACTORIZATION

The factored set of equations to be solved are:

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k}\right\}\left\{I + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\} \quad (D.1)$$

where for 2-D:

$$\left\{RHS^{n+1,k}\right\} =$$

$$-\frac{q^{n+1,k} - q^{n}}{\Delta\tau} - \left(\delta_\xi E^{n+1,k} + \delta_\zeta G^{n+1,k}\right) + \frac{1}{Re}\left(\delta_\xi R^{n+1,k} + \delta_\zeta T^{n+1,k}\right) \quad (D.2)$$

Eq. (D.1) is solved by performing two sweeps. First, a sweep in the ξ direction:

$$\left\{I + \Delta\tau\delta_\xi A^{n+1,k}\right\}\left\{\Delta q^{*}\right\} = \Delta\tau\left\{RHS^{n+1,k}\right\} \quad (D.3)$$

where $\{\Delta q^*\}$ is a temporary vector.

The next sweep is in the ζ direction:

$$\left\{I + \Delta\tau\delta_\zeta C^{n+1,k}\right\}\left\{\Delta q^{n+1,k}\right\} = \left\{\Delta q^*\right\} \qquad \text{(D.4)}$$

These two sweeps each require the inversion of a tridiagonal block matrix, which is computationally more efficient than the solution of the original pentadiagonal block matrix.

In 3-D, the same procedure is used at each spanwise station, with the RHS containing the explicit spanwise terms.

Since central differencing is used for the spatial derivatives, each block consists of a 4x4 matrix in 2-D, and a 5x5 matrix in 3-D. Eqs. (D.3) and (D.4) are solved by the block LU decomposition method.

APPENDIX E

ARTIFICIAL VISCOSITY

When a central differencing method is used to solve a non-linear PDE, numerical errors cause small oscillations in the solution to appear. At a given time level, the numerical solution may be written as:

$$q^n_{numerical} = q^n_{exact} + \sum \varepsilon \sin(\omega x) \qquad \text{(E.1)}$$

As this numerical solution is used to compute the flow field at successive time levels, the error in the solution in turn causes new errors at higher frequencies (i.e., the error at a frequency w will operate on itself, causing a new error component at a frequency of 2w). These new errors in turn cause even higher frequency errors at the next time level, and thus low frequency errors move up through the frequency band until the highest frequency that the grid can resolve is reached. At this point, the high frequency errors manifest themselves as low frequency errors again, and the cycle repeats and grows until the accuracy of the entire solution is destroyed.

An artificial viscosity model is implemented in order to damp out these numerical oscillations in order to prevent these errors from growing. A blended second and fourth order explicit dissipation is used, combined with an implicit second order dissipation term. This method uses fourth order dissipation except in regions containing shocks, where the second order dissipation terms become dominant.

This model is based on the numerical viscosity model proposed by Jameson, Turkel, and Schmidt and modified by Swanson and Turkel (Ref. 19).

The governing equation with the artificial viscosity model added may be written as follows:

$$\left\{I + \left(\Delta\tau\delta_\xi A^{n+1,k} + \varepsilon_I D_{I_\xi}\right)\right\}\left\{I + \left(\Delta\tau\delta_\zeta C^{n+1,k} + \varepsilon_I D_{I_\zeta}\right)\right\}\left\{\Delta q^{n+1,k}\right\} = \Delta\tau\left\{\left(RHS^{n+1,k}\right) - \varepsilon_E D_E\right\} \qquad \text{(E.2)}$$

where D is the dissipation terms, and the subscripts I and E refers to implicit and explicit terms respectively. The coefficients e_E and e_I are used to scale the magnitude of the dissipation terms. Usually, e_E is chosen to be 1.0 and e_I is 2.0.

The implicit dissipation terms are written:

$$D_{I_\xi} = \frac{1}{J_{ij}}\left(\lambda_{\xi_{ij}} \nabla_\xi \Delta_\xi J_{ij}\right)$$
$$D_{I_\zeta} = \frac{1}{J_{ij}}\left(\lambda_{\zeta_{ij}} \nabla_\zeta \Delta_\zeta J_{ij}\right) \qquad \text{(E.3)}$$

where:

$$\lambda_\xi = |U| + a\sqrt{\xi_x^2 + \xi_z^2}$$
$$\lambda_\zeta = |W| + a\sqrt{\zeta_x^2 + \zeta_z^2} \qquad \text{(2-D)}$$

$$\lambda_\xi = |U| + a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2} \quad \text{(3-D)}$$
$$\lambda_\zeta = |W| + a\sqrt{\zeta_x^2 + \zeta_y^2 + \zeta_z^2} \qquad \text{(E.4)}$$

Note that λ_ξ is the largest eigenvalue of the flux matrix **A**, and λ_ζ is the largest eigenvalue of the flux matrix **C**. The differencing operators are defined:

$$\Delta_\xi J_{ij} = J_{i+1,j} - J_{i,j}$$
$$\nabla_\xi J_{ij} = J_{i,j} - J_{i-1,j} \qquad \text{(E.5)}$$

The forward and backward differencing operators Δ and ∇ are defined in a similar way in the ζ direction.

The explicit dissipation terms are more complicated. In the present model, the explicit artificial viscosity is broken up into two terms:

$$D_E = D_{E_\xi} + D_{E_\zeta} \qquad \text{(E.6)}$$

The explicit dissipation term in the ξ-direction is defined as:

$$D_{E_\xi} = -\nabla_\xi\left[\left(\frac{\lambda_{\xi_{i+1,j}}}{J_{i+1,j}} + \frac{\lambda_{\xi_{i,j}}}{J_{i,j}}\right)\left(\varepsilon_{i,j}^{(2)}\Delta_\xi q_{i,j}^n\right)\right] + \nabla_\xi\Delta_\xi\left[\left(\frac{\lambda_{\xi_{i,j}}}{J_{i,j}}\right)\varepsilon_{i,j}^{(4)}\nabla_\xi\Delta_\xi q_{i,j}^n\right] \qquad \text{(E.7)}$$

where:

$$\varepsilon_{i,j}^{(2)} = k^{(2)}\max(\sigma_{i-1,j}, \sigma_{i,j}, \sigma_{i+1,j})$$
$$\sigma_{i,j} = \frac{|p_{i+1,j} - 2p_{i,j} + p_{i-1,j}|}{|p_{i+1,j} + 2p_{i,j} + p_{i-1,j}|}$$
$$\varepsilon_{i,j}^{(4)} = \max(0, k^{(4)} - \varepsilon_{i,j}^{(2)}) \qquad \text{(E.8)}$$

Usually, $k^{(2)} = 0.25$ and $k^{(4)} = 0.01$. The term λ_ξ is defined in Eq. (E.4).

The explicit dissipation term in the ζ-direction (and the η direction for 3-D) is defined in a similar manner.

Near the boundaries of the computational grid, the fourth order dissipation term poses a problem. With the model given by Eq. (E.6), information is needed at five nodes for the computation of the explicit dissipation term. When the grid point adjacent to the boundary is reached, there are only four nodes available. Therefore, special expressions must be developed for use next to the boundaries.

To accomplish this, ghost points are defined outside of the computational domain (on the i=0 line and the i = imax+1 line). Values on these lines are defined by extrapolating from the interior of the domain:

$$q_{0,j} = 2q_{1,j} - q_{2,j} \qquad \text{(E.9)}$$

and similarly for the i=imax+1 line.

This provides enough information to use Eq. (E.7) to compute the fourth order dissipation term.

APPENDIX F

THE BALDWIN-LOMAX TURBULENCE MODEL

The Baldwin-Lomax model is an algebraic turbulence model which computes an eddy viscosity which can be added to the molecular viscosity to obtain an effective viscosity:

$$\mu_{effective} = \mu_{molecular} + \mu_{turb} \qquad (F.1)$$

A two-layer model is defined for the eddy viscosity. Near the wall, the eddy viscosity is proportional to the local vorticity and the distance from the wall multiplied by the Prandtl-Van Driest damping factor. Thus,

$$\mu_{turb,inner} = \rho L^2 |\omega| \qquad (F.2)$$

where

$$|\omega| = \sqrt{\left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2} \quad \text{(2-D)}$$

$$|\omega| = \sqrt{\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)^2} \quad \text{(3-D)}$$

$$L = \kappa z D \tag{F.3}$$

Here, k = 0.4 is the von Karman constant, z is the normal distance from the wall, and D is the Prandtl-Van Driest damping factor, which smoothly goes to zero at the wall:

$$D = 1 - \exp\left(\frac{-z\rho_w \tau_w}{26\mu_w}\right) \tag{F.4}$$

The subscript 'w' refers to values at the wall.

The outer layer uses this formula to compute eddy viscosity away from the body:

$$\mu_{turb,outer} = K_c C_{cp} \rho F_{wake} F_{kleb}(z) \tag{F.5}$$

where K_c = 0.0168 is the Clauser's constant, and C_{cp} = 1.6 is an empirical constant. Also,

$$F_{wake} = \min\left(z_{max} F_{max}, \frac{0.25 z_{max} U_{dif}^2}{F_{max}}\right) \tag{F.6}$$

and z_{max} and F_{max} are at the maximum of

$$F(z) = z|\omega|\left\{1 - \exp\left(\frac{-z\rho_w \tau_w}{26\mu_w}\right)\right\} \tag{F.7}$$

In the wake region, the exponential is dropped.

The last definitions are:

$$U_{dif} = \left(\sqrt{u^2 + w^2}\right)_{max} - \left(\sqrt{u^2 + w^2}\right)_{min} \quad \text{(2-D)}$$

$$U_{dif} = \left(\sqrt{u^2 + v^2 + w^2}\right)_{max} - \left(\sqrt{u^2 + v^2 + w^2}\right)_{min} \quad \text{(3-D)}$$

$$F_{kleb} = \frac{1}{1 + 5.5\left(\frac{C_{kleb} z}{z_{max}}\right)^6} \qquad \text{(F.8)}$$

where F_{kleb} is the Klebanoff intermittency correction, which smoothly reduces the eddy viscosity to zero in the far field. The Klebanoff constant is C_{kleb}, which is taken to be 0.3.

The switch between the inner layer and outer layer eddy viscosity occurs at z = $z_{crossover}$. This crossover value of z is defined as the closest point to the wall where the inner and outer eddy viscosities are equal. Thus,

$$\mu_{turb} = \begin{cases} \mu_{turb,\,inner} \text{ for } z \le z_{crossover} \\ \mu_{turb,\,outer} \text{ for } z \ge z_{crossover} \end{cases} \qquad \text{(F.9)}$$

APPENDIX G

INITIAL AND BOUNDARY CONDITIONS

The Navier-Stokes equations are parabolic in time and elliptic in space. To solve a parabolic equation, a marching procedure is used. This means that the solution is marched in time from some meaningful initial solution until the desired time level, or a steady-state solution, is reached. An impulsive start from rest is the initial condition set in this investigation. Thus, the flow properties off the solid body are set to freestream conditions for the initial solution.

Elliptic equations require the values of the variables on the boundaries to be specified. This means that meaningful values of the flow properties must be assigned on the boundaries of the computational grid. These boundary values effectively define the problem that is being solved, so care should be taken in their selection.

Outer Boundaries

For external flows, the outer boundary of the computational grid is placed far from the body. For unsteady flow problems, the outer boundary is divided into inflow and outflow boundaries. On inflow boundaries, the freestream values of all the flow variables are used. On outflow boundaries, ρ, u, v, and p are specified by extrapolating from the values from adjacent grid points. Then e is calculated from these.

In 2-D, different boundary conditions can be used for steady flow problems. For steady inviscid flows, 1-D Riemann invariants are used with a circulation correction. The Riemann invariants are defined:

$$R^- = V_n - \frac{2a}{\gamma - 1}$$
$$R^+ = V_n + \frac{2a}{\gamma - 1} \qquad \text{(G.1)}$$

where V_n is the local normal velocity:

$$V_n = \frac{\zeta_x u + \zeta_y w}{\sqrt{\zeta_x^2 + \zeta_z^2}} \qquad \text{(G.2)}$$

The local tangential velocity is given as:

$$V_t = \frac{\xi_z u - \xi_x w}{\sqrt{\xi_x^2 + \xi_z^2}} \qquad \text{(G.3)}$$

Eq. (G.1) and (G.3) give three equations for specifying the four flow variables. The fourth equation, which is for the entropy parameter S, is given by:

$$S = \ln\left(\frac{p}{\rho^\gamma}\right) \qquad \text{(G.4)}$$

The procedure used is as follows. First, the boundary is defined as an inflow or an outflow boundary, using Eq. (G.2). On an inflow boundary, $V_n < 0$ and the Riemann invariant R^- is constant along waves that run upstream (from the interior of the grid).

Thus, R^-, V_t, and S are computed using freestream values and the other invariant, R^+, is extrapolated from the interior. These are used to update the flow variables ρ, u, v, and e. On an outflow boundary, $V_n > 0$, and R^+, V_t, and S are extrapolated from the interior while R^- is specified using freestream values.

Another outer boundary condition used for steady flows is a circulation correction. Lift is generated by circulation about an airfoil. Since the outer boundary is far from the airfoil, the airfoil can be modeled as a point vortex with circulation G, located at the quarter-chord. G is defined:

$$\Gamma = \frac{1}{2} M_\infty^2 c\, C_l \qquad \text{(G.5)}$$

where C_l is the lift coefficient, c is the chord length of the airfoil, and M_∞ is the freestream Mach number.

The velocity perturbation caused by this vortex is:

$$V_p = \frac{\left(\sqrt{1 - M_\infty^2}\right)\Gamma}{2\pi r\left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)} \qquad \text{(G.6)}$$

where r is the radius measured from the airfoil quarter-chord and θ the angle measured from the airfoil centerline, and α is the angle of attack of the airfoil.

The corrected boundary velocities due to this vortex are:

$$u_b = u_\infty + V_p \sin\theta$$
$$w_b = w_\infty + V_p \cos\theta \qquad \text{(G.7)}$$

C-2

It can be seen that the perturbation velocity decreases linearly with distance; if the outer boundary is far enough away, this effect is negligible. If this correction is used, however, the outer boundary may be moved much closer to the airfoil with no loss in accuracy.

To satisfy the condition for constant enthalpy, the speed of sound must also be corrected using the freestream enthalpy H_∞:

$$a_b^2 = (\gamma - 1)\left[H_\infty - \frac{1}{2}\left(u_b^2 + w_b^2\right)\right] \qquad \text{(G.8)}$$

The last outer boundary is the wake cut aft of the airfoil. Flow properties are simply averaged across this cut, since the variation is smooth in this region.

In 3-D, there are two additional outer boundaries at the root of the wing (j=1), and at the far-field off the tip (j=jmax). At the root, a symmetry condition (spanwise derivative = 0) is used:

$$q_{i,1,k} = \frac{4q_{i,2,k} - q_{i,3,k}}{3} \qquad \text{(G.9)}$$

At the (j=jmax) station, the unsteady boundary conditions described above are used.

Solid Boundaries

The boundary conditions on the wall are dictated by whether the flow is viscous or inviscid. In both analyses, the fluid has no normal velocity component with respect to

the body. This gives rise to the 'no penetration' condition on the contravariant velocity W, which is normal to the body:

$$W = \zeta_t + u\zeta_x + w\zeta_z = 0 \quad \text{(2-D)}$$
$$W = \zeta_t + u\zeta_x + v\zeta_y + w\zeta_z = 0 \quad \text{(3-D)} \qquad \text{(G.9)}$$

The tangential contravariant velocity U for inviscid flow is extrapolated from the points adjacent to the body. For viscous flows, U is set to zero (no slip). The physical velocities u and w can then be determined from the contravariant velocities.

The density on the body is extrapolated from the adjacent two points in the normal direction using:

$$\rho_{i,1} = 2\rho_{i,2} - \rho_{i,3} \qquad \text{(G.10)}$$

The surface pressure satisfies the condition:

$$\frac{\partial p}{\partial \zeta} = 0 \qquad \text{(G.11)}$$

which is numerically approximated by:

$$p_{i,1} = \frac{(4p_{i,2} - p_{i,3})}{3} \qquad \text{(G.12)}$$

Using these conditions, the total energy may be determined.

For simplicity, the boundary conditions are calculated explicitly at the end of each call to the Newton iteration routine.

REFERENCES

1. Borland, C.J. and Rizzetta, D., "Nonlinear Transonic Flutter Analysis," AIAA Paper 81-0608-CP, AIAA Dynamic Specialists Conference, 1981.

2. Rizzetta, D.P. and Borland, C., "Numerical Solution of Unsteady Transonic Flow over Wings with Viscous-Inviscid Interaction", AIAA Paper 82-0352, January 1982.

3. Batina, J. T., "Unsteady Transonic Algorithm Improvements for Realistic Aircraft Applications", AIAA Paper 88-0105, January 1988.

4. Sankar, L.N., Malone, J.B., and Tassa, Y., "An Implicit Conservative Algorithm for Steady and Unsteady Three-Dimensional Transonic Potential Flows", AIAA Paper 81-1016-CP, June 1981.

5. Malone, J.B. and Sankar, L.N., "Application of a Three-Dimensional Steady and Unsteady Full Potential Method for Transonic Flow Computations", AFWAL-TR-84-3011, Flight Dynamics Laboratory, Wright Patterson Air Force Base, Dayton, Ohio, 1984.

6. Shankar, V., Ide, H., Gorski, J. and Osher, S., "A Fast, Time-Accurate Unsteady Full Potential Scheme", AIAA Paper 85-1512-CP, July 1985.

7. Pulliam, T.H., and Steger, J.L., "Implicit Finite Difference Simulations of Three-Dimensional Compressible Flow", AIAA Journal, Vol. 18, 1980.

8. Batina, J.T., "Unsteady Euler Solutions Using Unstructured Dynamic Meshes", AIAA Paper No. 89-0115, January 1989.

9. Sankar, L. N. and Tang, W., "Numerical Solution of Unsteady Viscous Flow past Rotor Sections", AIAA Paper 85-0129.

10. Wake, B. E. and Sankar, L. N., "Solution of the Navier-Stokes Equations for the Flow About a Rotor Blade", Journal of the American Helicopter Society, April 1989.

11. Rai, M. M., "Navier-Stokes Simulations of Rotor-Stator Interaction Using Patched and Overlaid Grids", AIAA Paper 85-1519-CP, July 1985.

12. Gatlin, B. and Whitfield, D. L., "An Implicit Upwind Finite Volume Scheme for Solving the Three-Dimensional Thin-Layer Navier-Stokes Equations", AIAA Paper 87-1149-CP, June 1987.

13. Sankar, L. N. and Kwon, O. J., "Viscous Flow Simulation of Fighter Aircraft", AIAA Paper 91-0278, January 1991.

14. Saad, Y. and Schultz, M.H, "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems", SIAM J. Sci. Stat. Comp., Vol. 7, No. 3, 1986, pp.856-869.

15. Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes", AIAA Paper 85-1494-CP, 1985.

16. Venkatakrishnan, V. and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes", ICASE Report 91-40, May 1991.

17. Saad, Y. and Semeraro, B. D. , Application of Krylov Exponential Propagation to Fluid Dynamics Equations", AIAA Paper 91-1567-CP, 1991.

18. Beam, R. M. and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Vol. 16, No. 4, April, 1976.

19. Swanson, R. C. and Turkel, E., "Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations", AIAA Paper 87-1107-CP, June 1987.

20. Steger, J. L., "Implicit Finite Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries", AIAA Journal, Vol. 18, No. 2, pp. 159-167, Feb. 1980.

21. McAlister, K.W., Pucci, S.L., McCroskey, W.J., and Carr, L.W., "An Experimental Study of Dynamic Stall on Advanced Airfoil Section, Volume 2: Pressure and Force Data", NASA TM 84245, Sept. 1982.

22. Stagg, A.K., Cline, D.D., Shadid, J.N., and Carey, G.F., "A Performance Comparison of Massively Parallel Parabolized Navier-Stokes Solutions", AIAA Paper 93-0059, Jan. 1993.

23. Yoon, S. and Kwak, D., "Implicit Navier-Stokes Solver for Three Dimensional Compressible Flows," AIAA Journal, Vol. 30, No. 11, Nov. 1992.

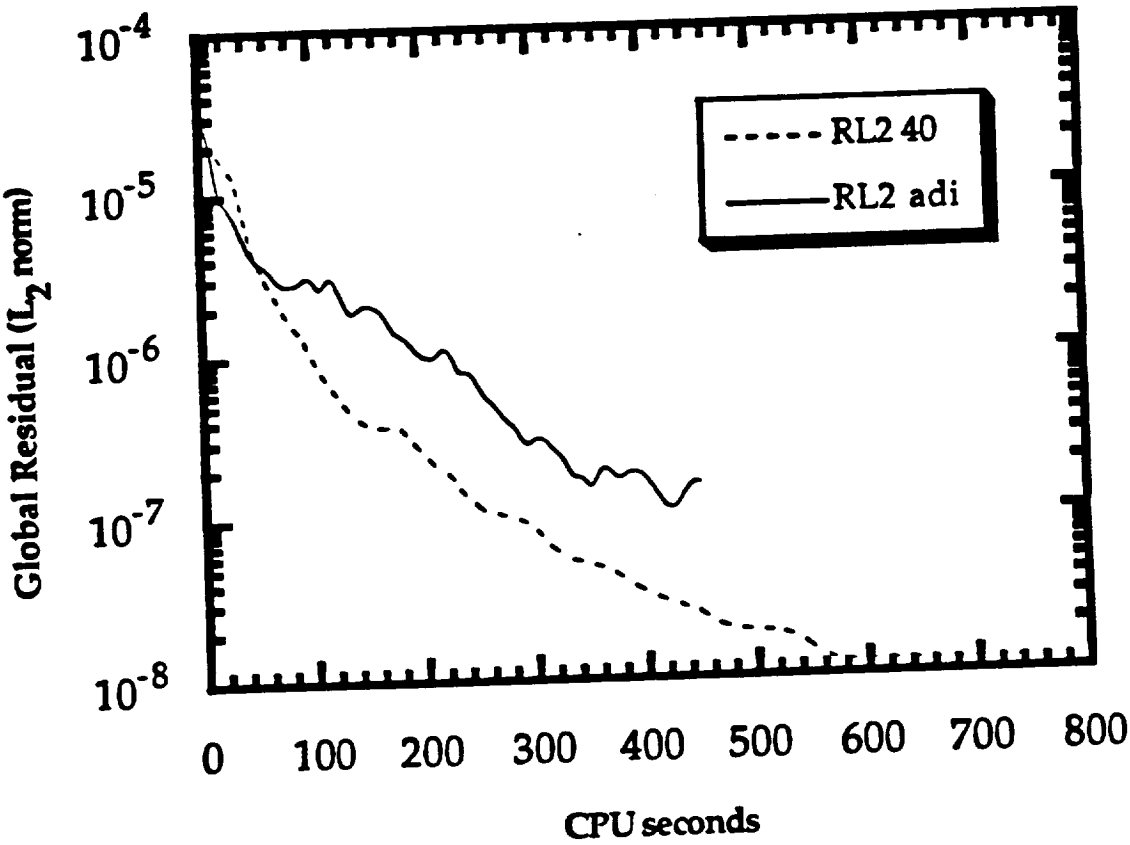

Figure 1
Comparison of the Global Residual History for the Calculation of a Steady Inviscid Transonic Flow about a NACA 0012 Airfoil ($M_\infty = 0.8; \alpha = 1.25°$)

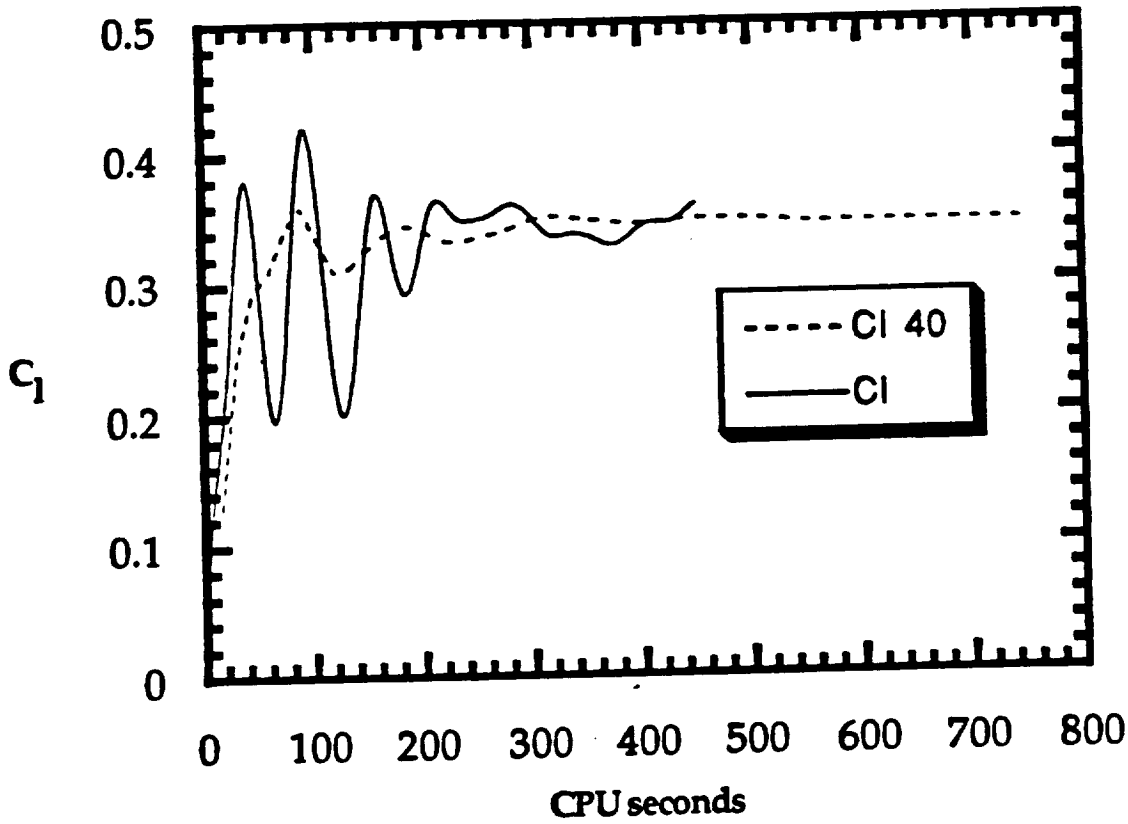

Figure 2
Comparison of the Lift Coefficient History for the Calculation of a Steady Inviscid Transonic Flow about a NACA 0012 Airfoil ($M_\infty = 0.8; \alpha = 1.25°$)

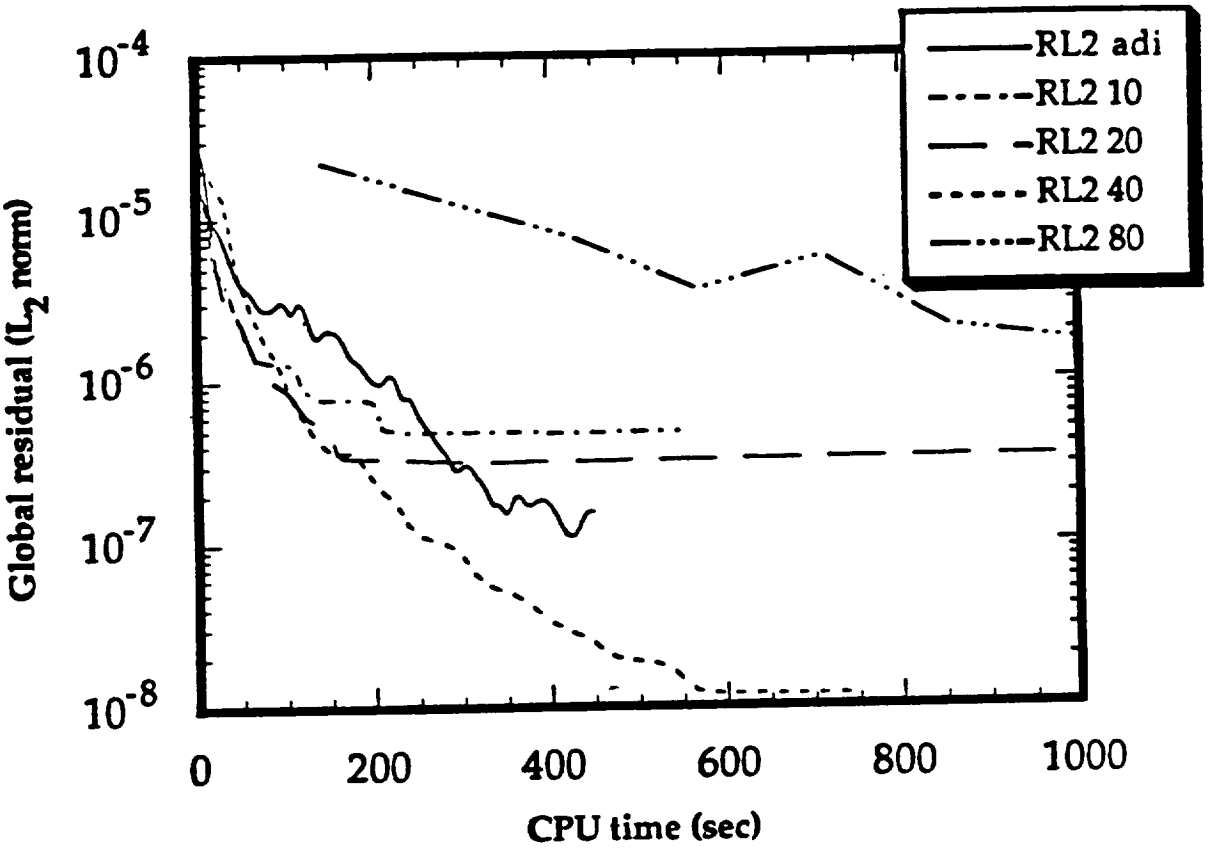

Figure 3
Comparison of the Global Residual History for Calculation of a Steady Inviscid Transonic Flow about a NACA 0012 Airfoil ($M_\infty = 0.8$; $\alpha = 1.25°$)

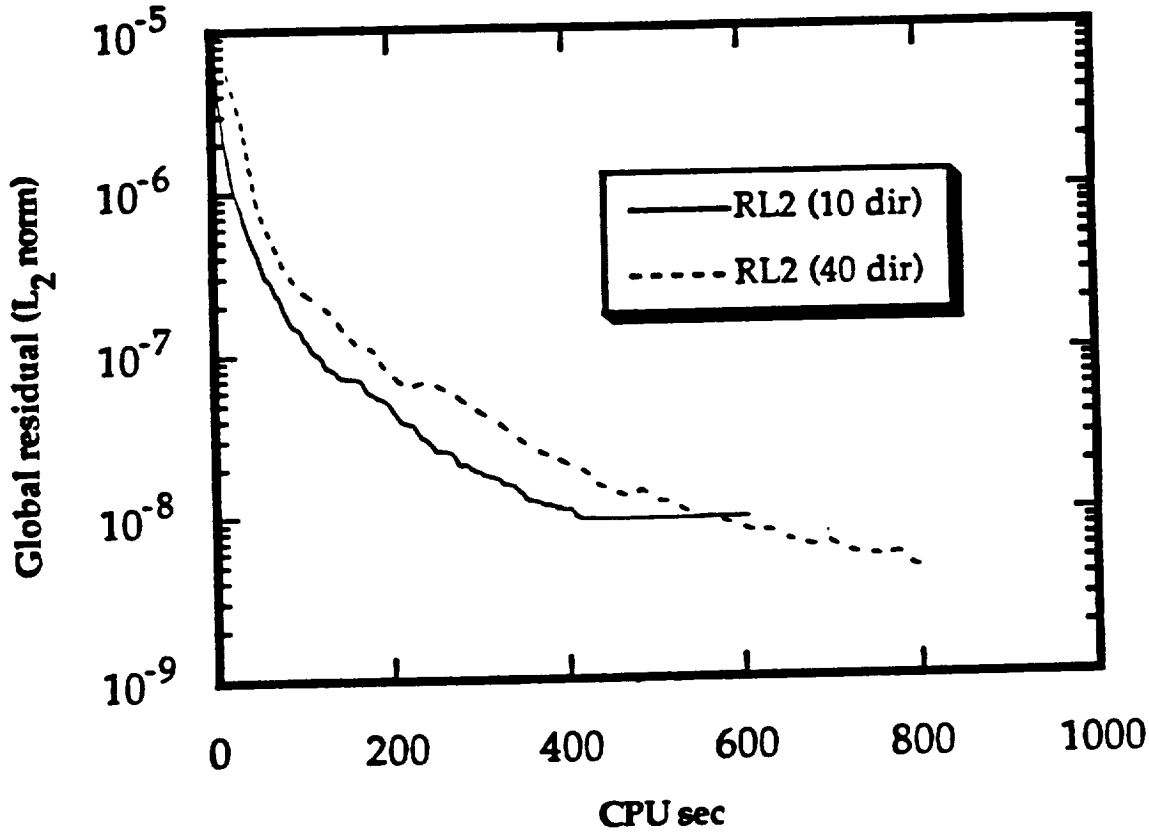

Figure 4
Comparison of the Global Residual History for the Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil
($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

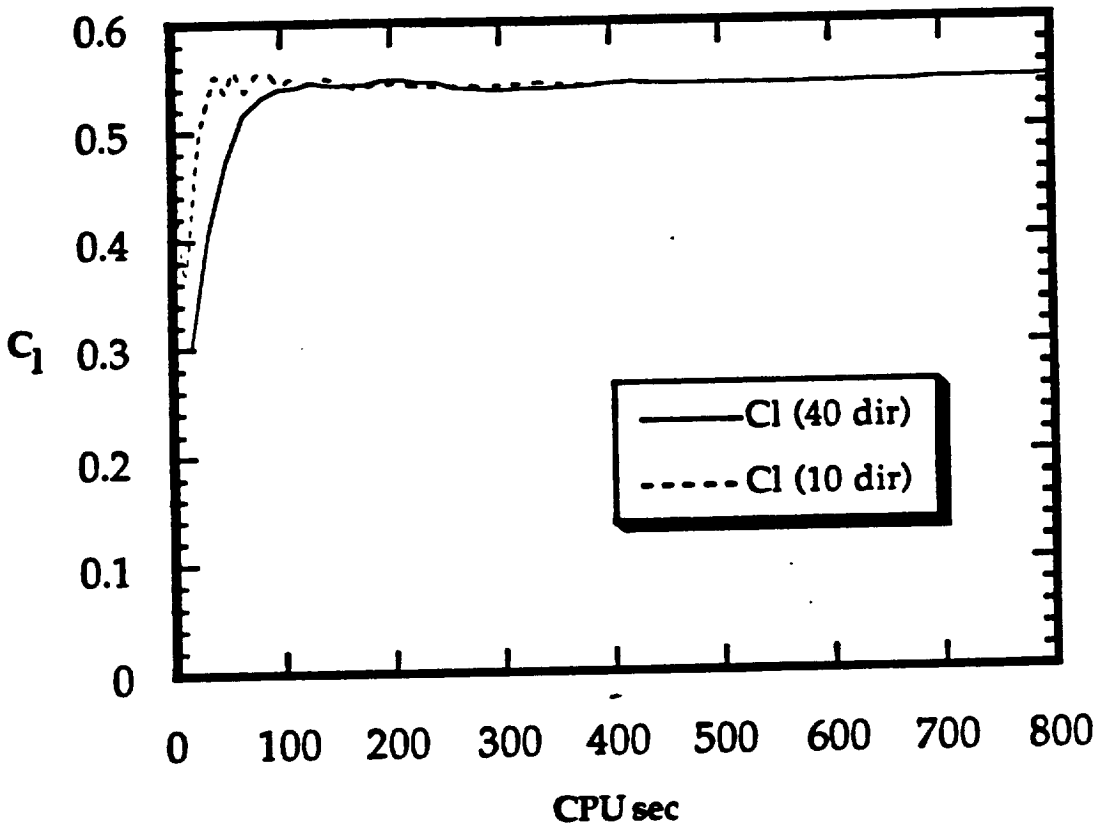

Figure 5
Comparison of the Lift Coefficient History for the Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil
($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

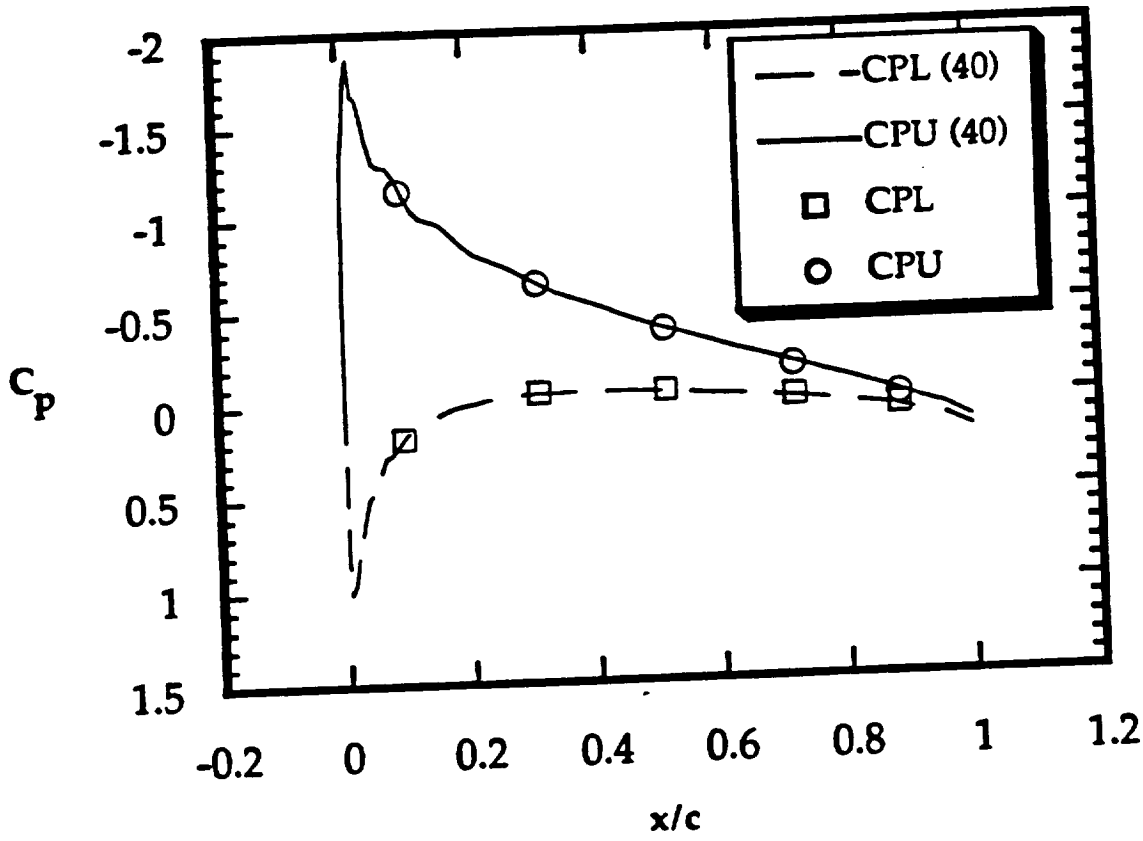

Figure 6
Comparison of the Pressure Coefficient for the Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil
($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

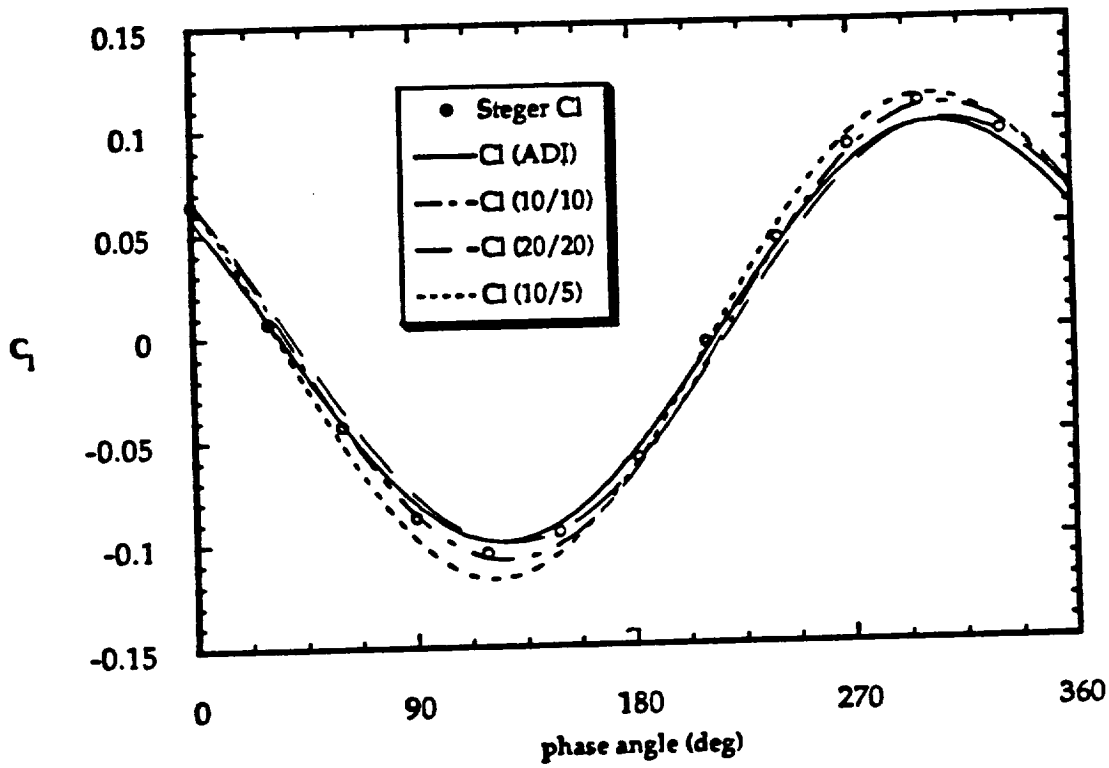

Figure 7
Effect of Time Step on GMRES Result for Lift Coefficient of a Plunging NACA 64-A010 Airfoil in Inviscid Transonic Flow ($M_\infty = 0.8$; $k = 0.2$)

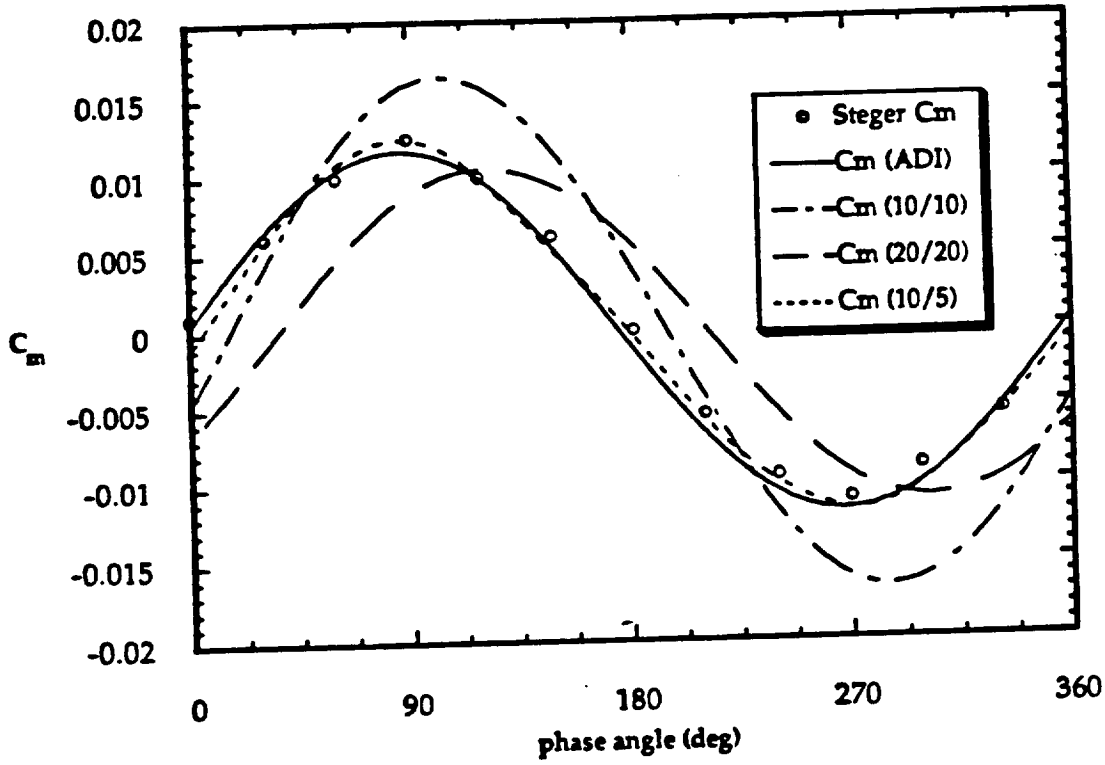

Figure 8
Effect of Time Step on GMRES Result for Moment Coefficient of a Plunging NACA 64-A010 Airfoil in Inviscid Transonic Flow ($M_\infty = 0.8$; $k = 0.2$)

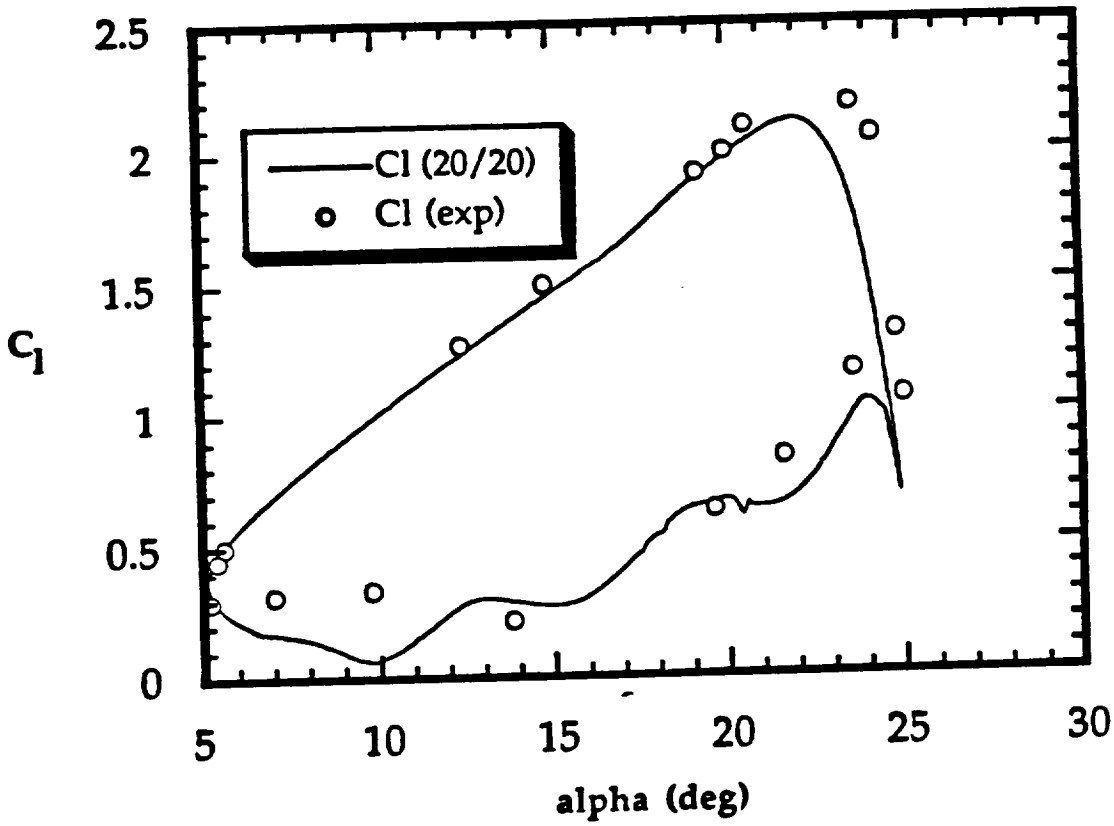

Figure 9
Comparison of GMRES (20/20) with Experimental Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

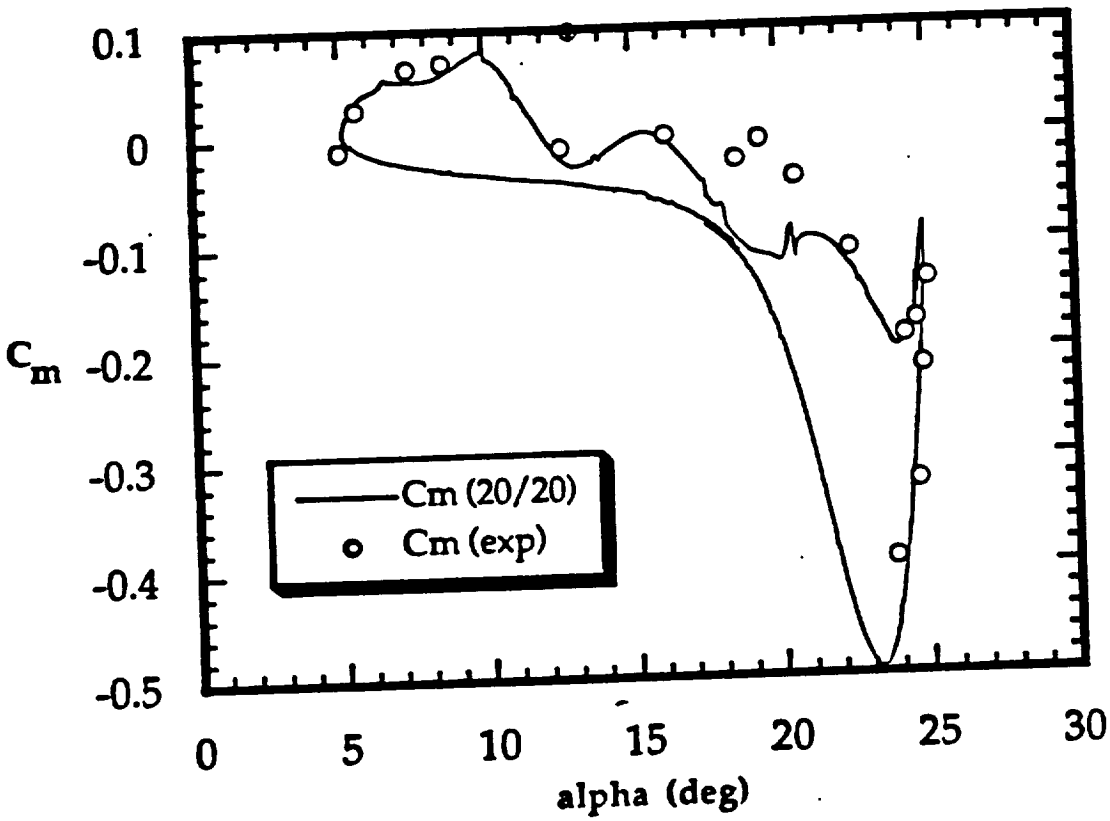

Figure 10
Comparison of GMRES (20/20) with Experimental Results for the Moment Coefficient of a Pitching NACA 0012 Airfoil ($M_\infty = 0.283$; k = 0.151; Re = 3,450,000)

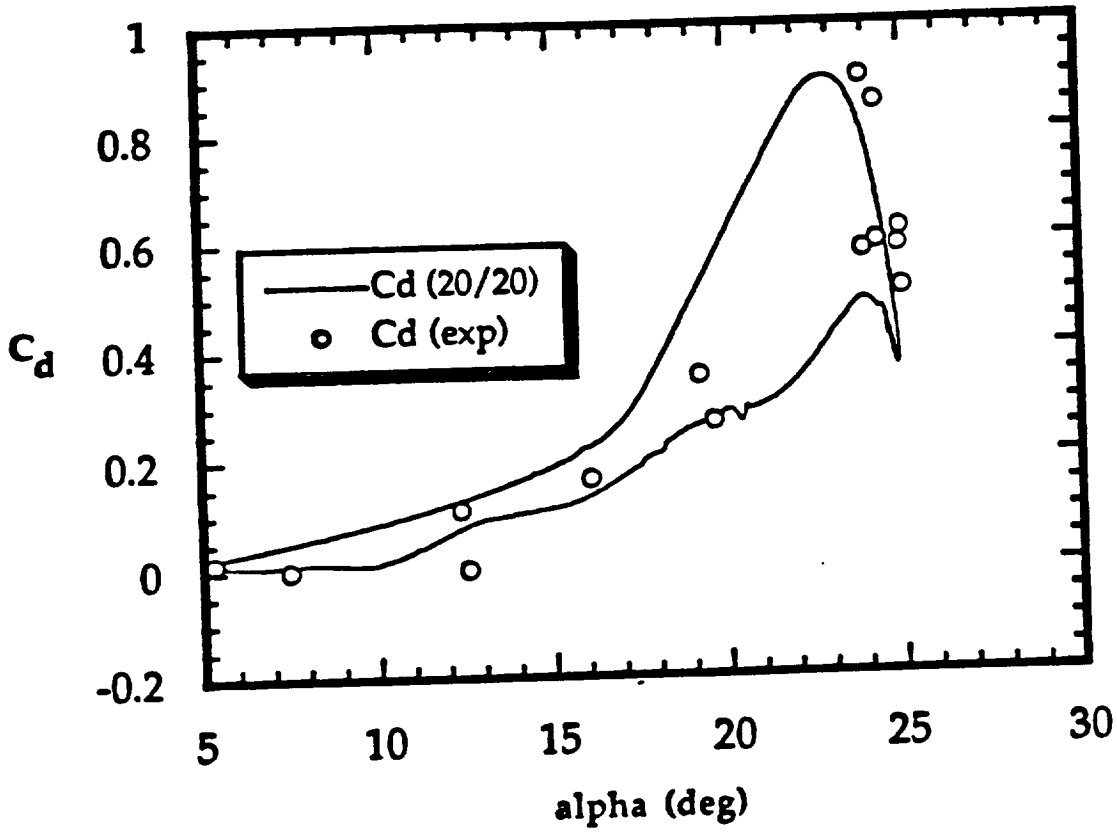

Figure 11
Comparison of GMRES (20/20) with Experimental Results for the Drag Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

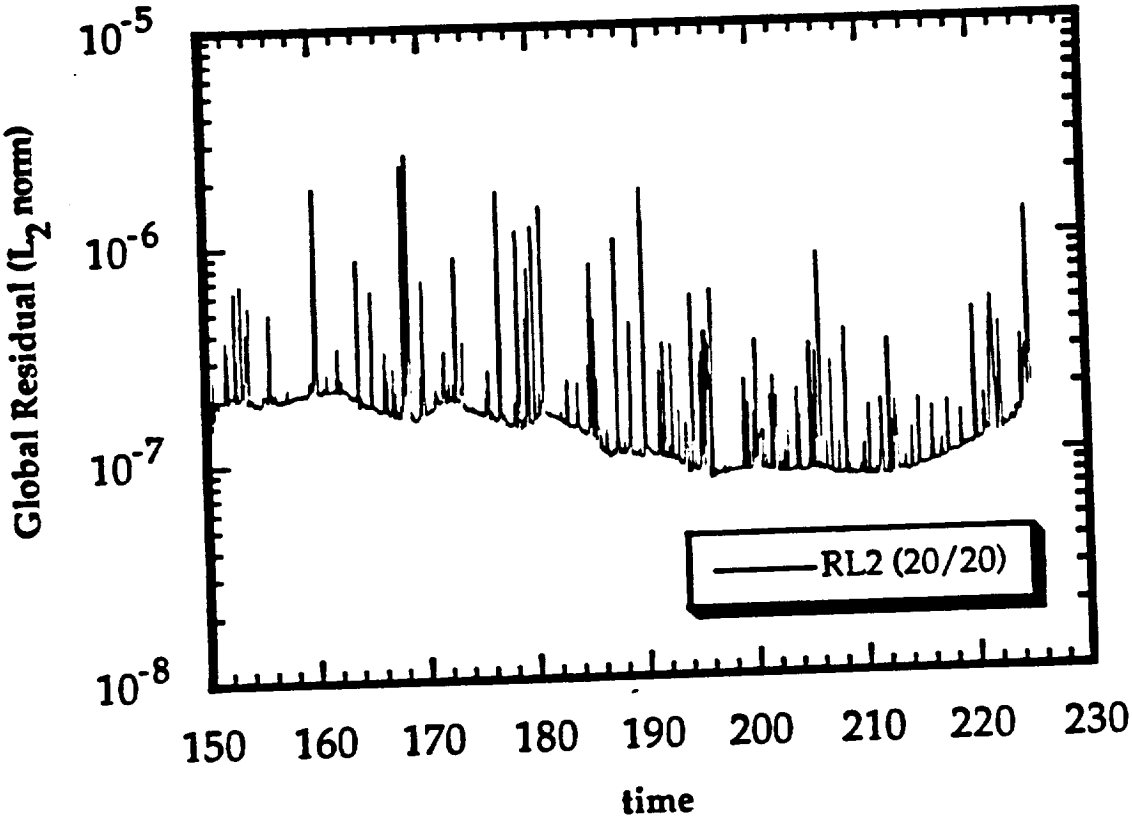

Figure 12
Global Residual History of a GMRES (20/20) Calculation of the Flow about a Pitching NACA 0012 Airfoil
($M_\infty = 0.283$; $k = 0.151$; $Re = 3{,}450{,}000$)

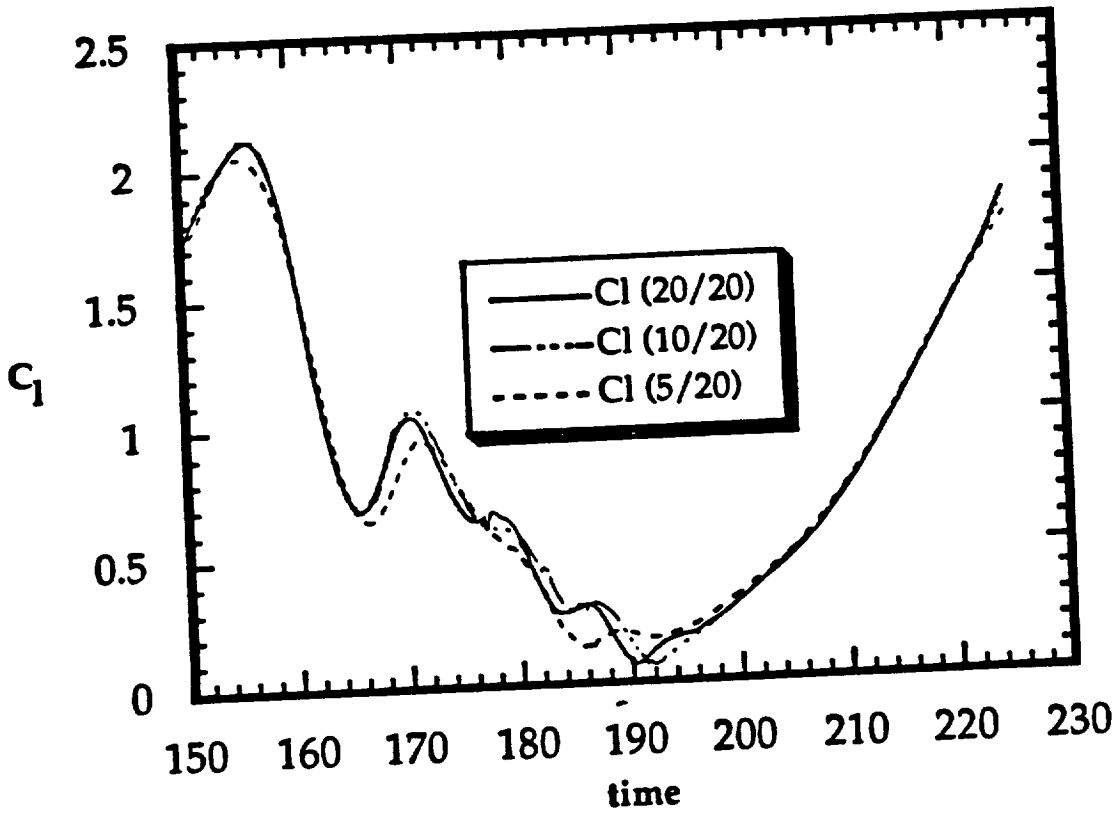

Figure 13
Effect of Directions on the GMRES (x/20) Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

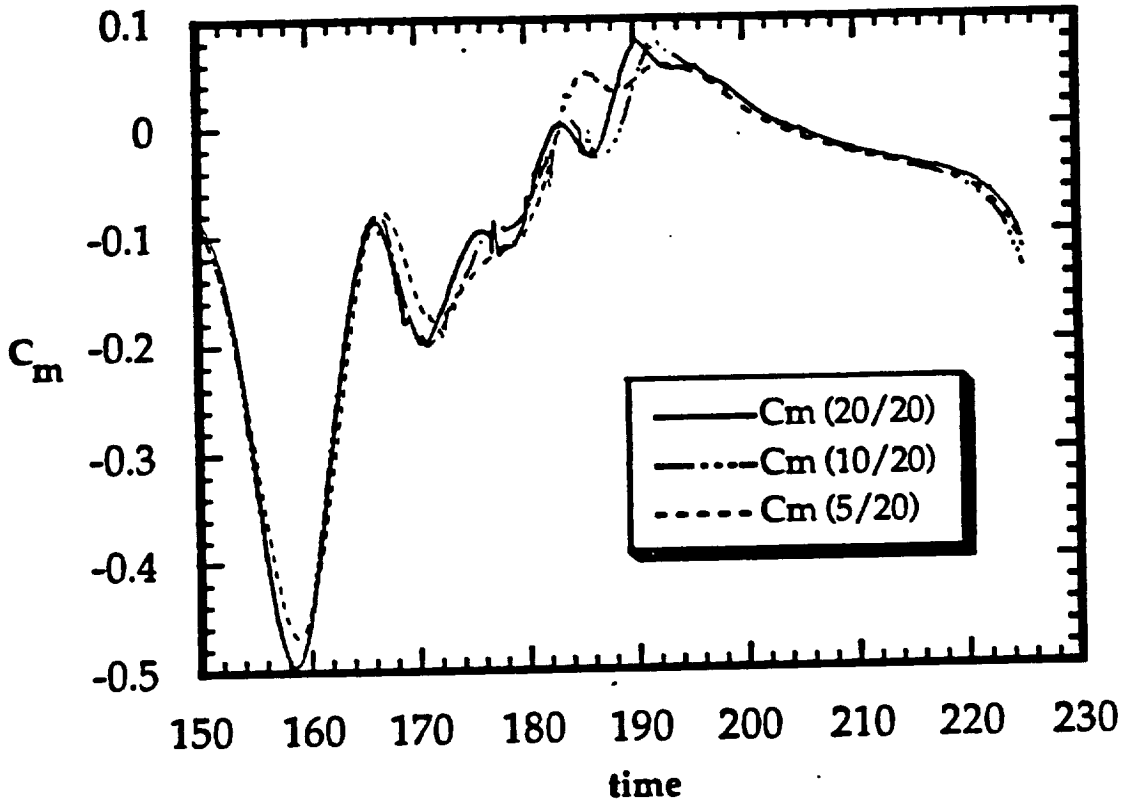

Figure 14
Effect of Directions on the GMRES (x/20) Results for the Moment Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

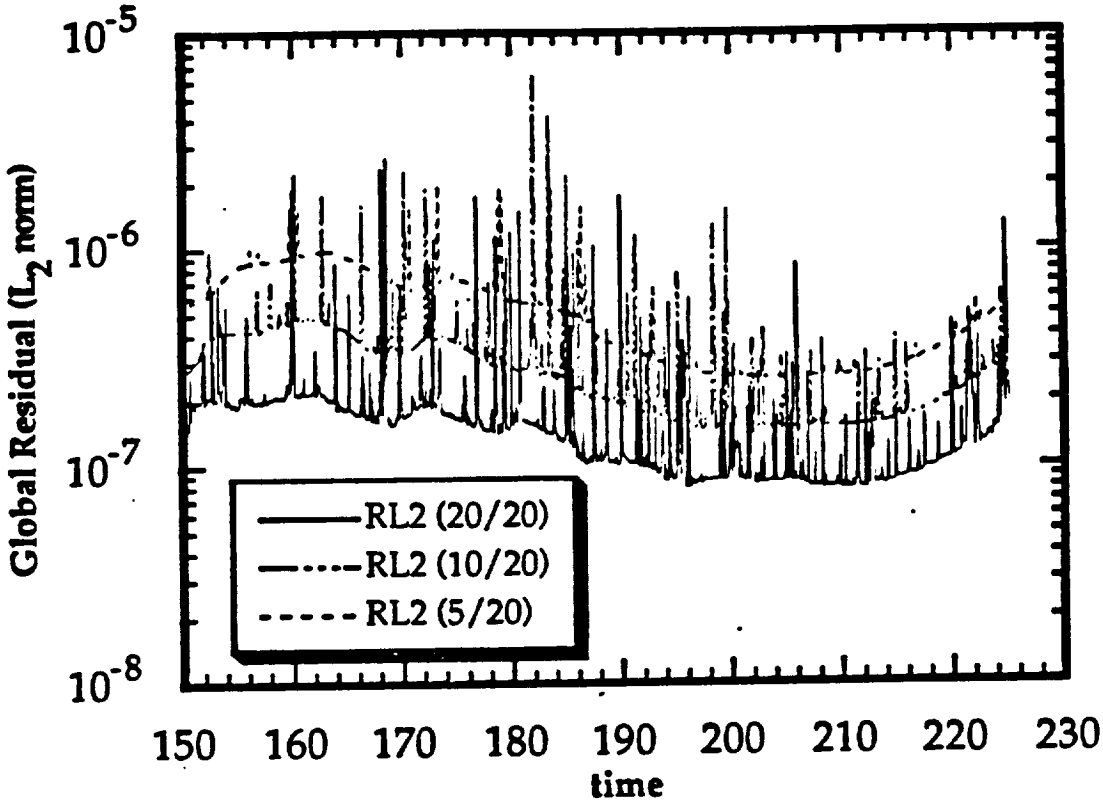

Figure 15
Effect of Directions on the GMRES (x/20) Results for the Global Residual of a Pitching NACA 0012 Airfoil Calculation (M_∞ = 0.283; k = 0.151; Re = 3,450,000)

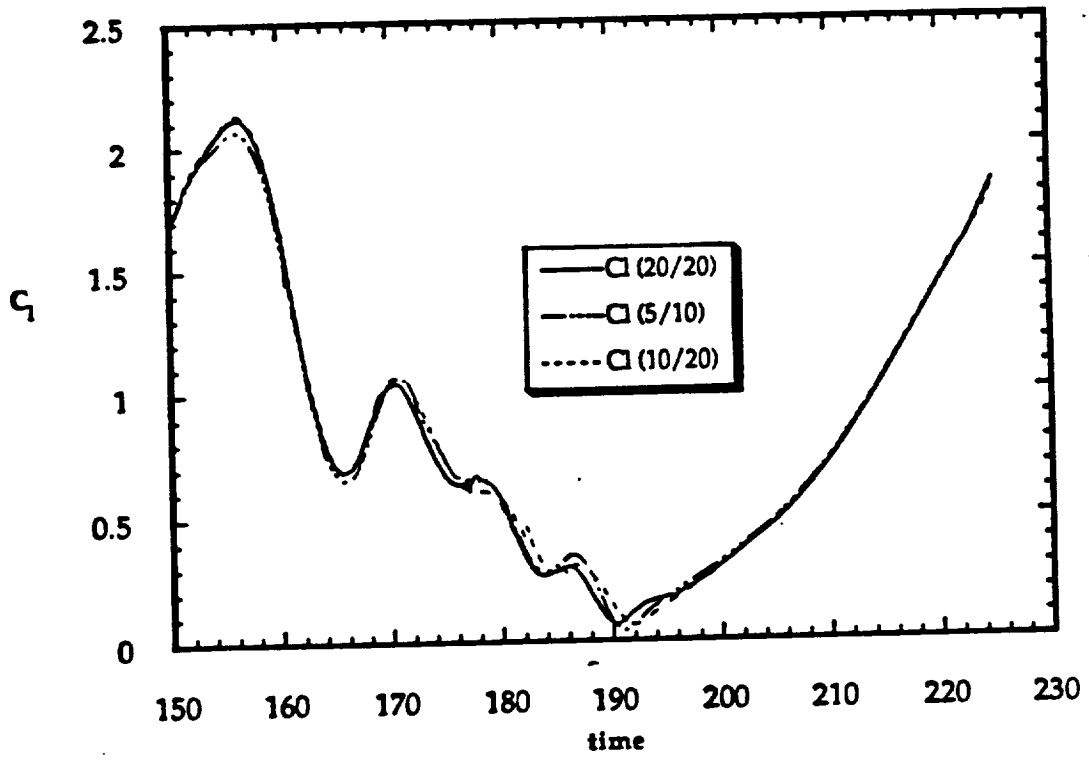

Figure 16
Comparison of GMRES (x/2x) Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

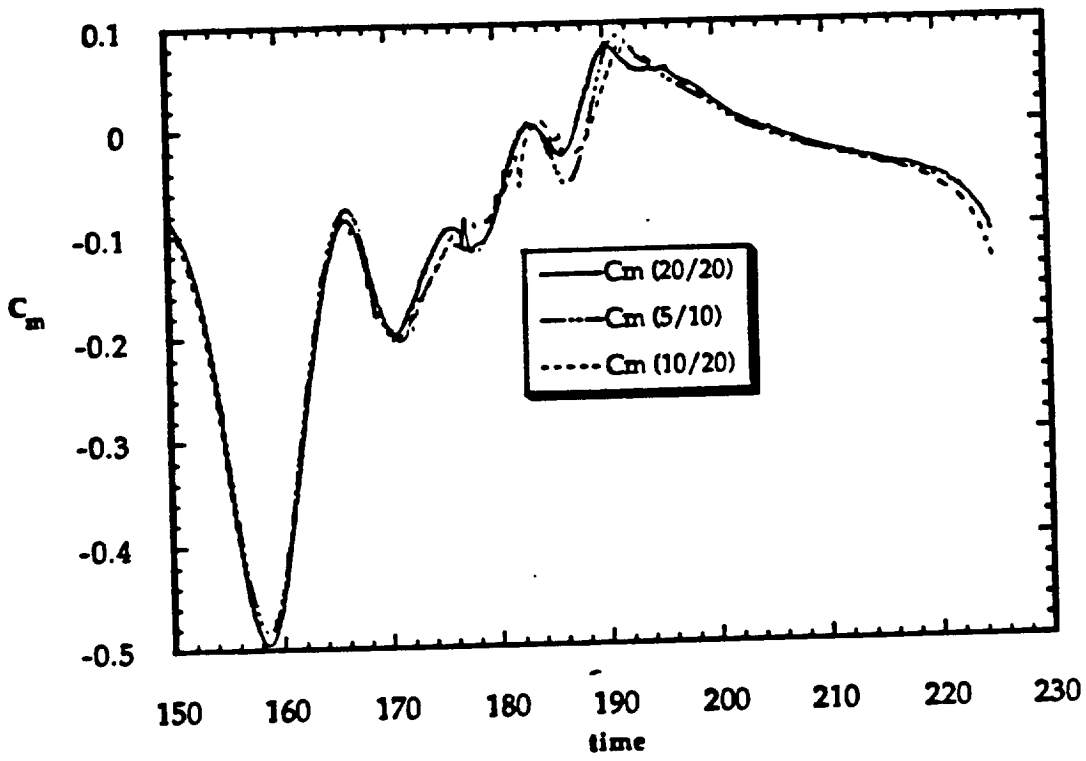

Figure 17
Comparison of GMRES (x/2x) Results for the Moment Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

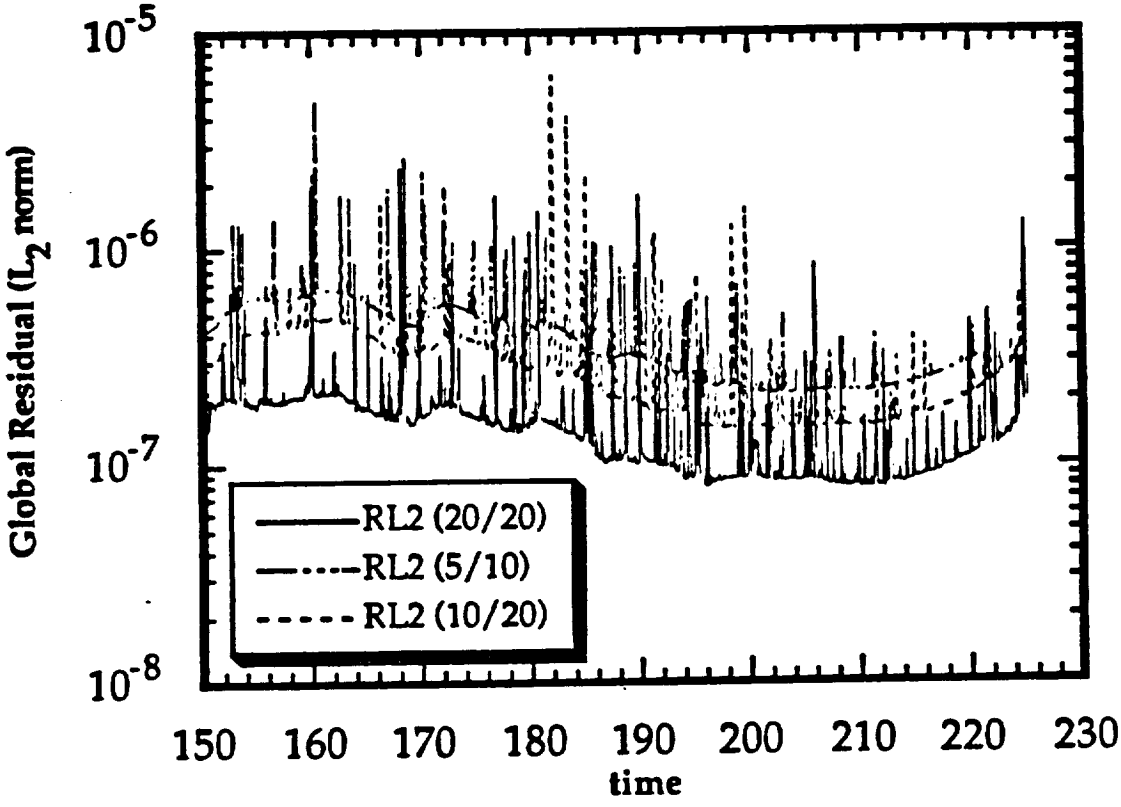

Figure 18
Comparison of GMRES (x/2x) Global Residual Histories for the Calculation of Flow about a Pitching NACA 0012 Airfoil (M_∞ = 0.283; k = 0.151; Re = 3,450,000)

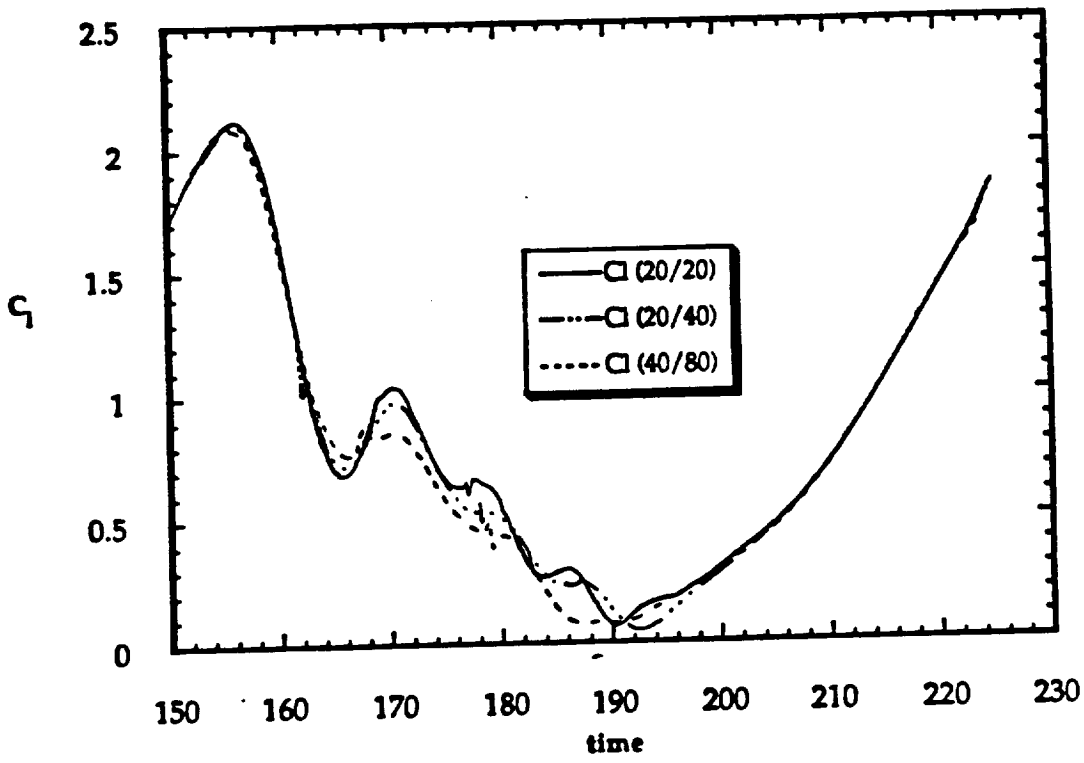

Figure 19
Comparison of GMRES (x/2x) Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

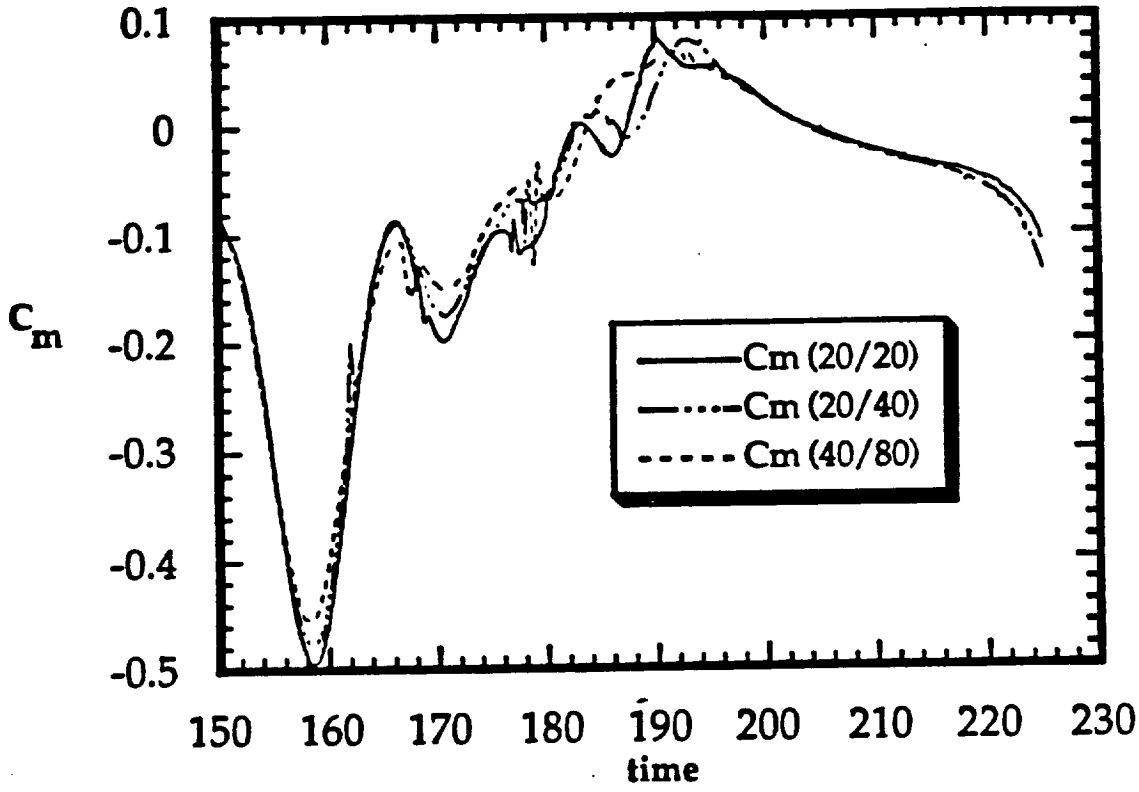

Figure 20
Comparison of GMRES (x/2x) Results for the Moment Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

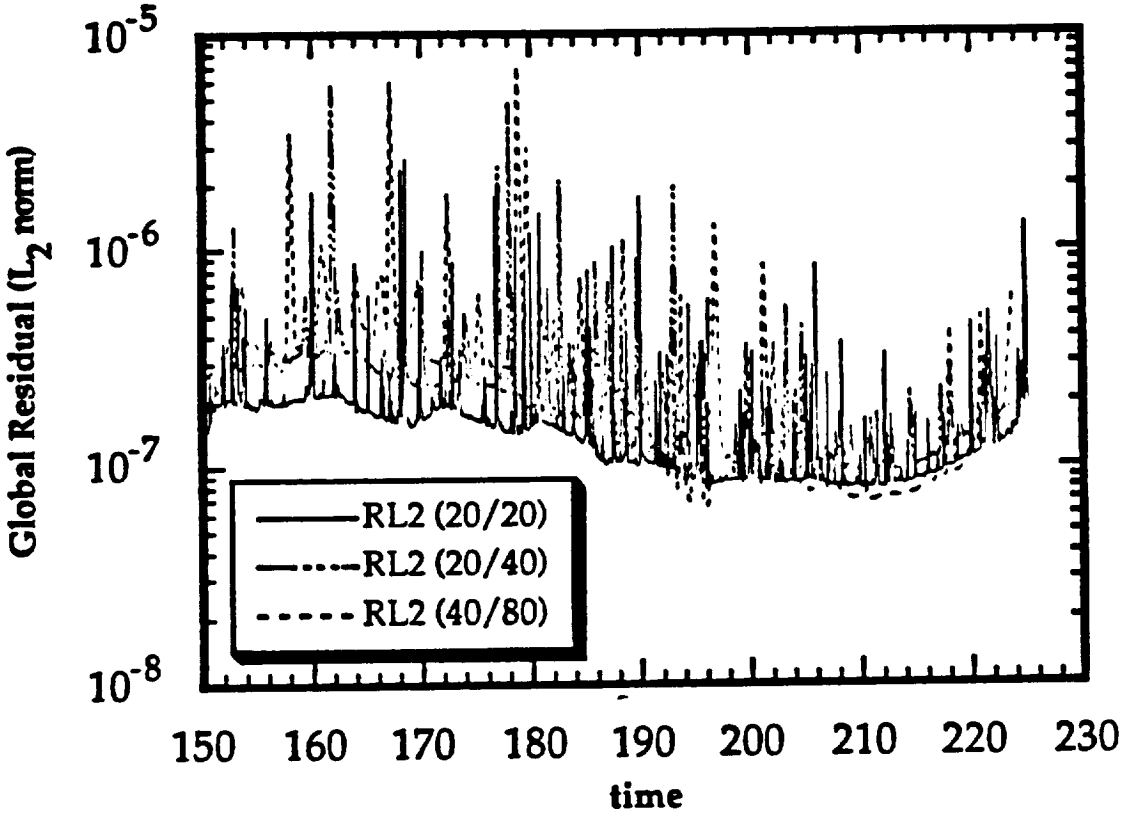

Figure 21
Comparison of GMRES (x/2x) Global Residual Histories for the Calculation of Flow about a Pitching NACA 0012 Airfoil (M_∞ = 0.283; k = 0.151; Re = 3,450,000)

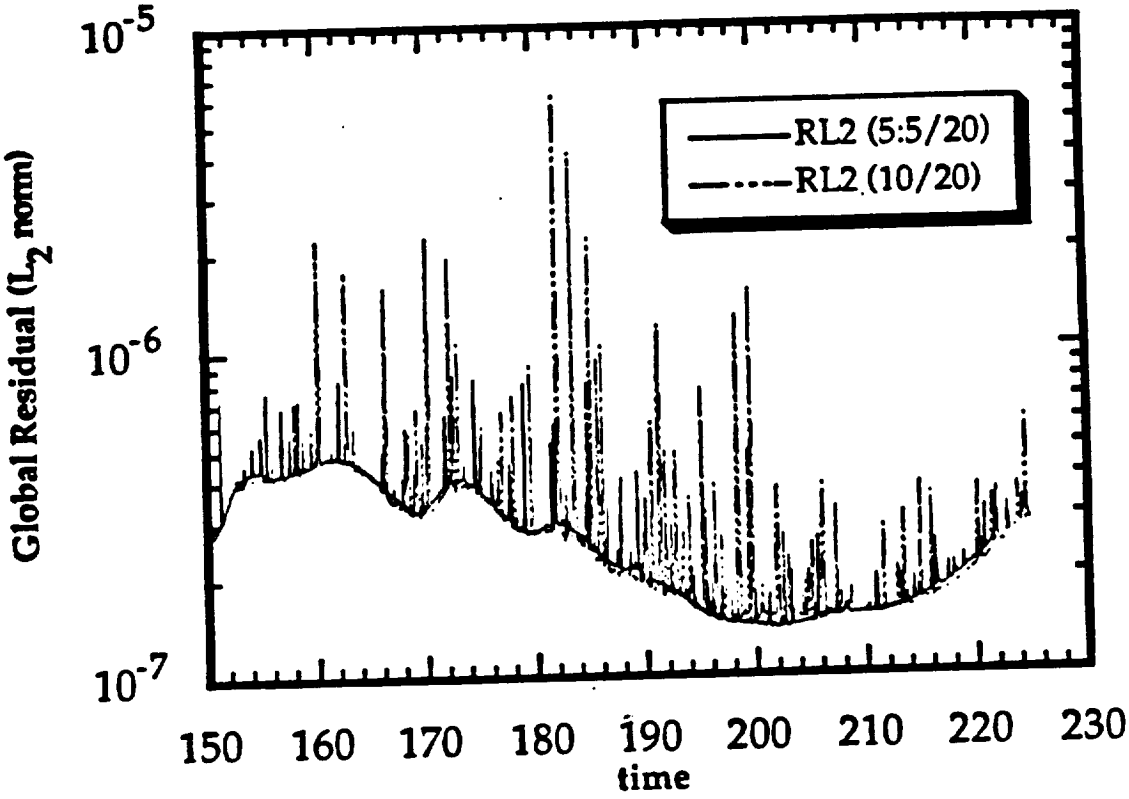

Figure 22
Restarted GMRES (5:5/20) Global Residual History for Flow about a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

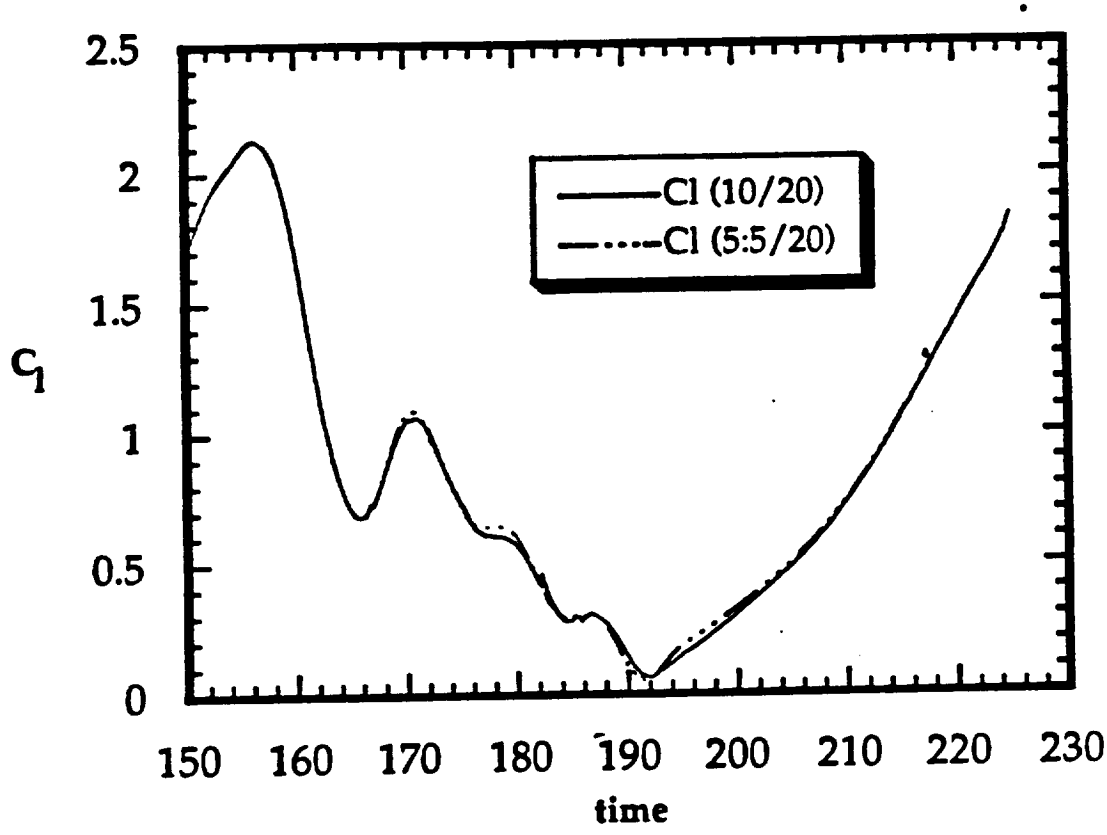

Figure 23
Restarted GMRES (5:5/20) Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

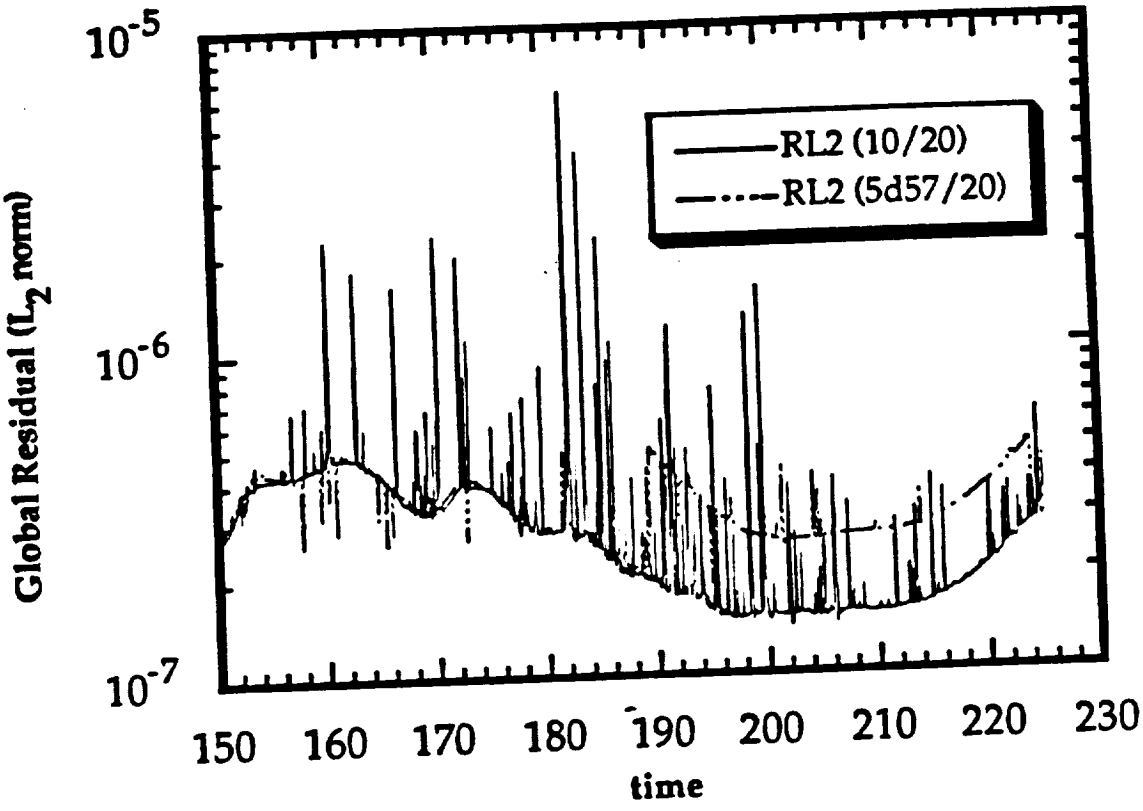

Figure 24
Dynamic Restart GMRES (5d57/20) Global Residual History for Flow about a Pitching NACA 0012 Airfoil ($M_\infty = 0.283$; $k = 0.151$; $Re = 3{,}450{,}000$)

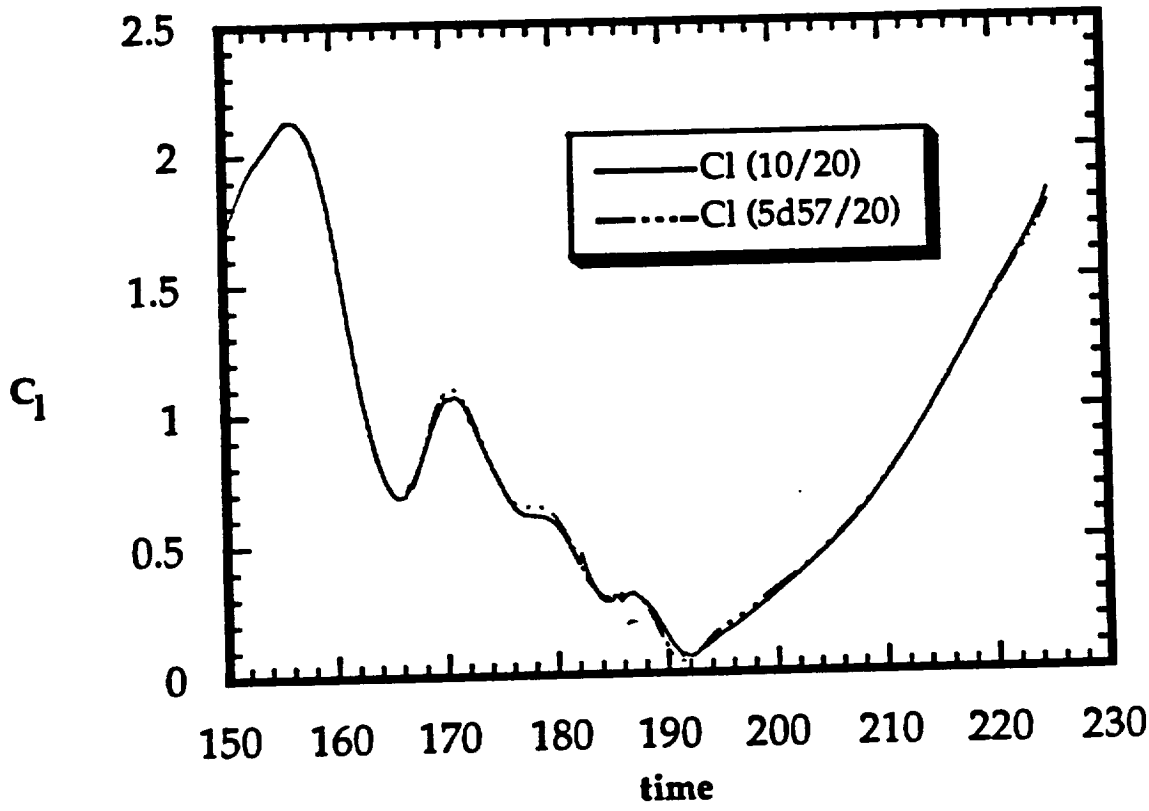

Figure 25
Dynamic Restart GMRES (5d57/20) Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

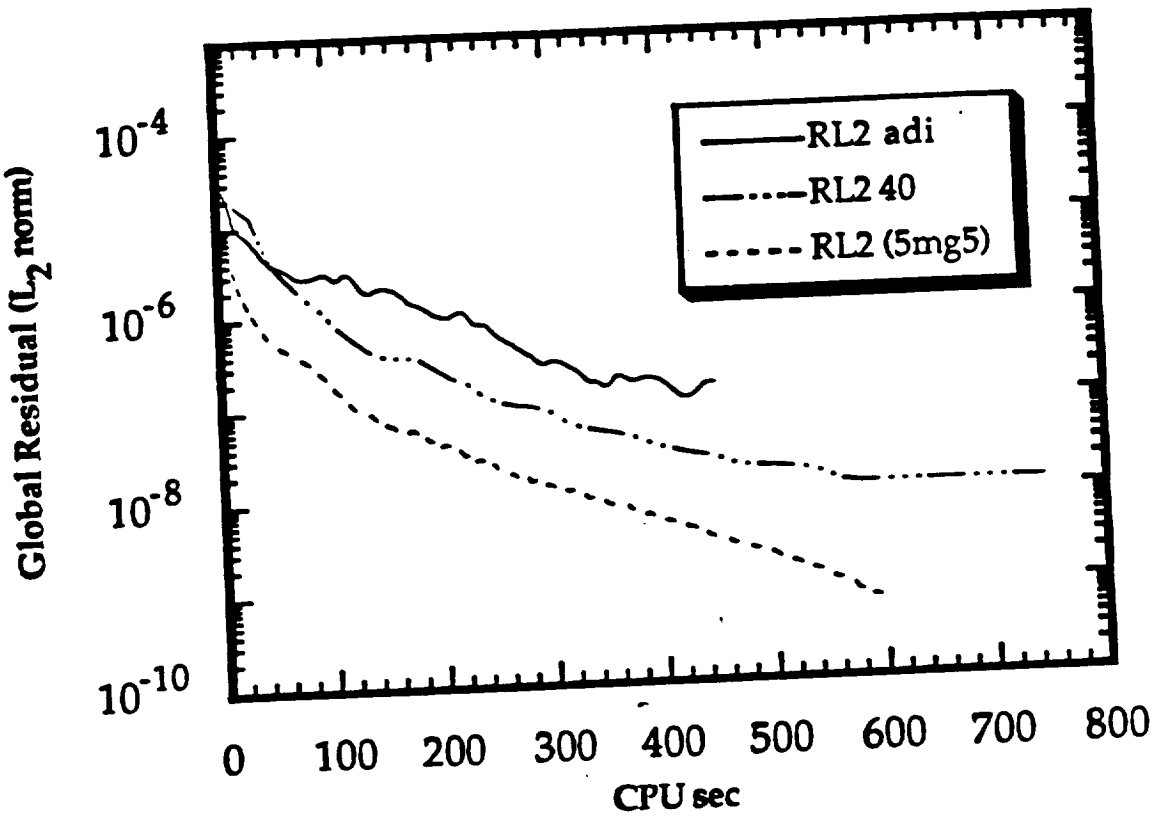

Figure 26
Comparison of the Multigrid Global Residual History for the Calculation of a Steady Inviscid Transonic Flow about a NACA 0012 Airfoil ($M_\infty = 0.8$; $\alpha = 1.25°$)

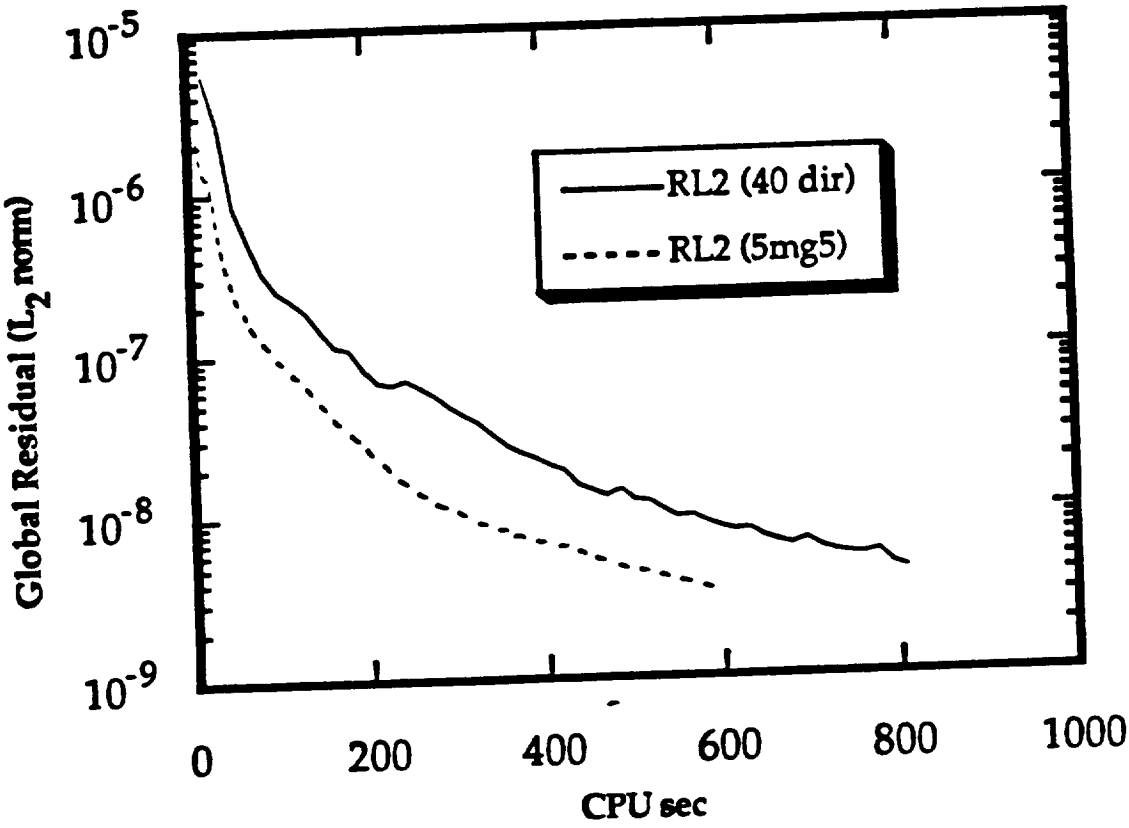

Figure 27
Comparison of the Multigrid Global Residual History for the Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil ($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

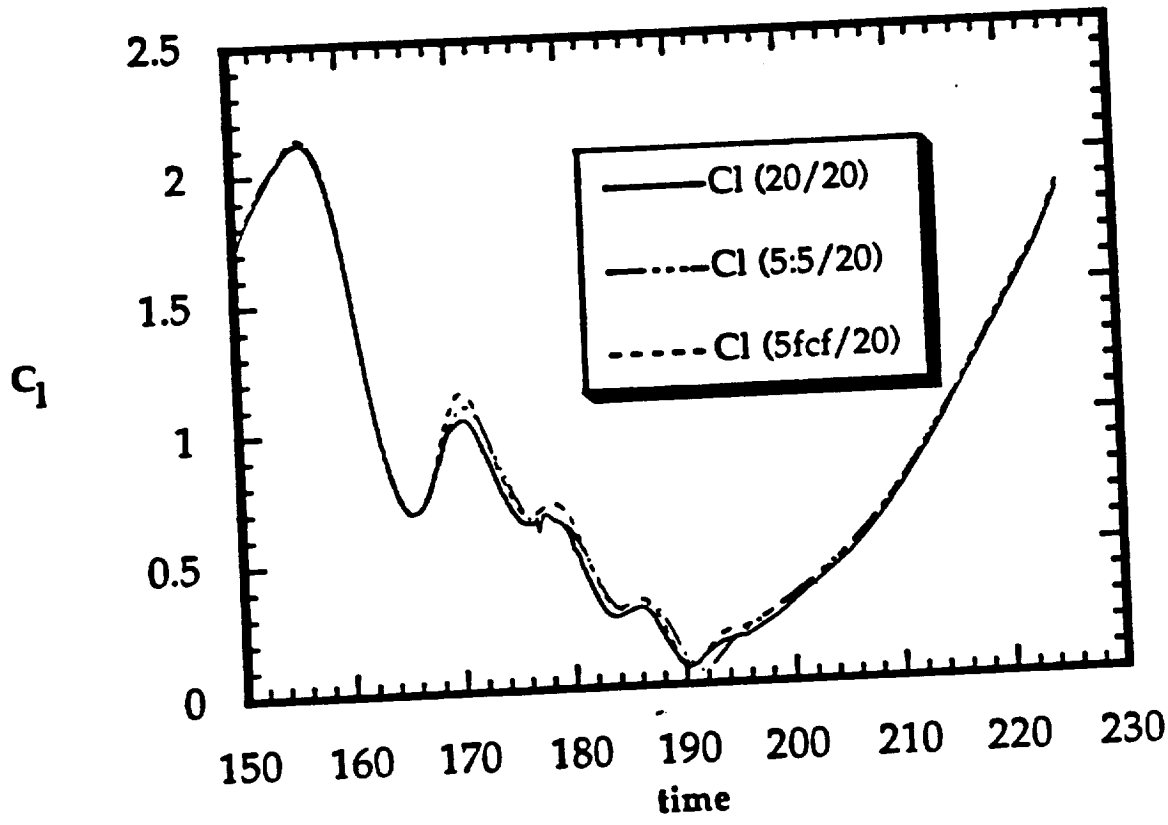

Figure 28
Multigrid GMRES Results for the Lift Coefficient of a Pitching NACA 0012 Airfoil
($M_\infty = 0.283$; k = 0.151; Re = 3,450,000)

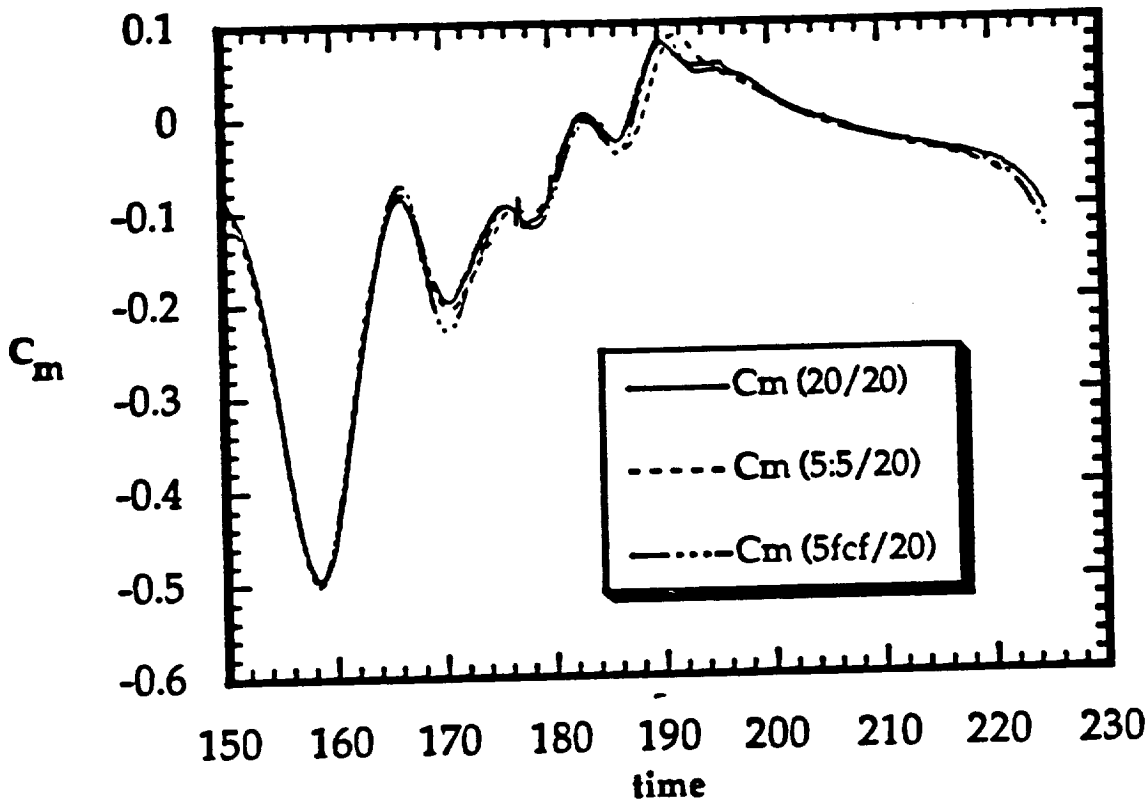

Figure 29
Multigrid GMRES Results for the Moment Coefficient of a Pitching NACA 0012 Airfoil
(M_∞ = 0.283; k = 0.151; Re = 3,450,000)

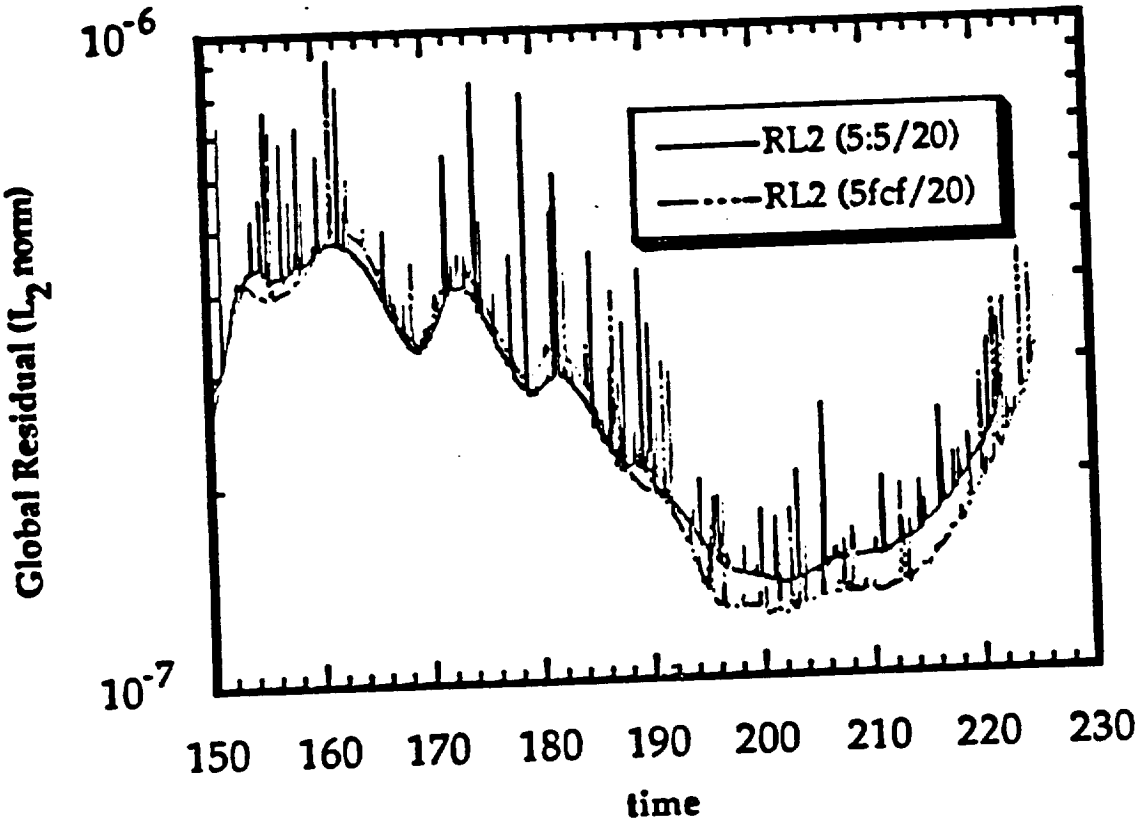

Figure 30
Multigrid GMRES Global Residual History for the Calculation of Flow about a Pitching NACA 0012 Airfoil ($M_\infty = 0.283$; $k = 0.151$; $Re = 3{,}450{,}000$)

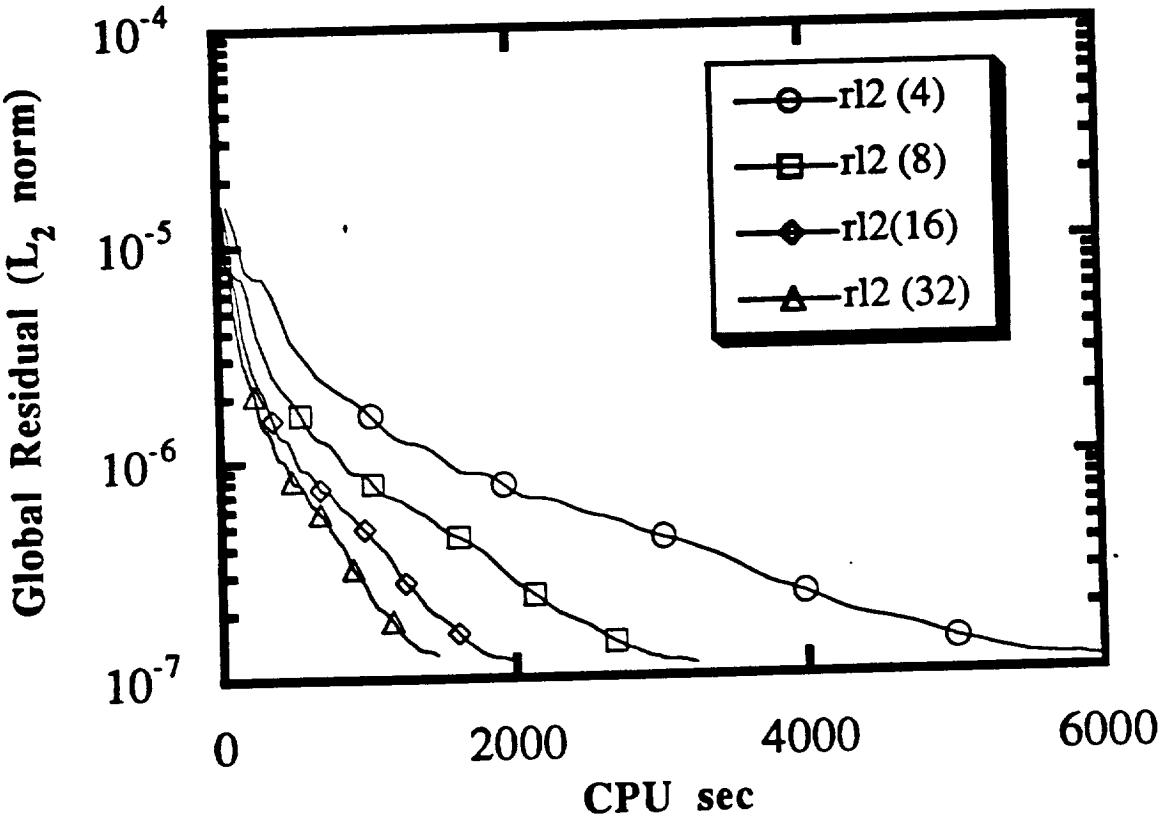

Figure 31
Comparison of the Global Residual History for the Parallel Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil ($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

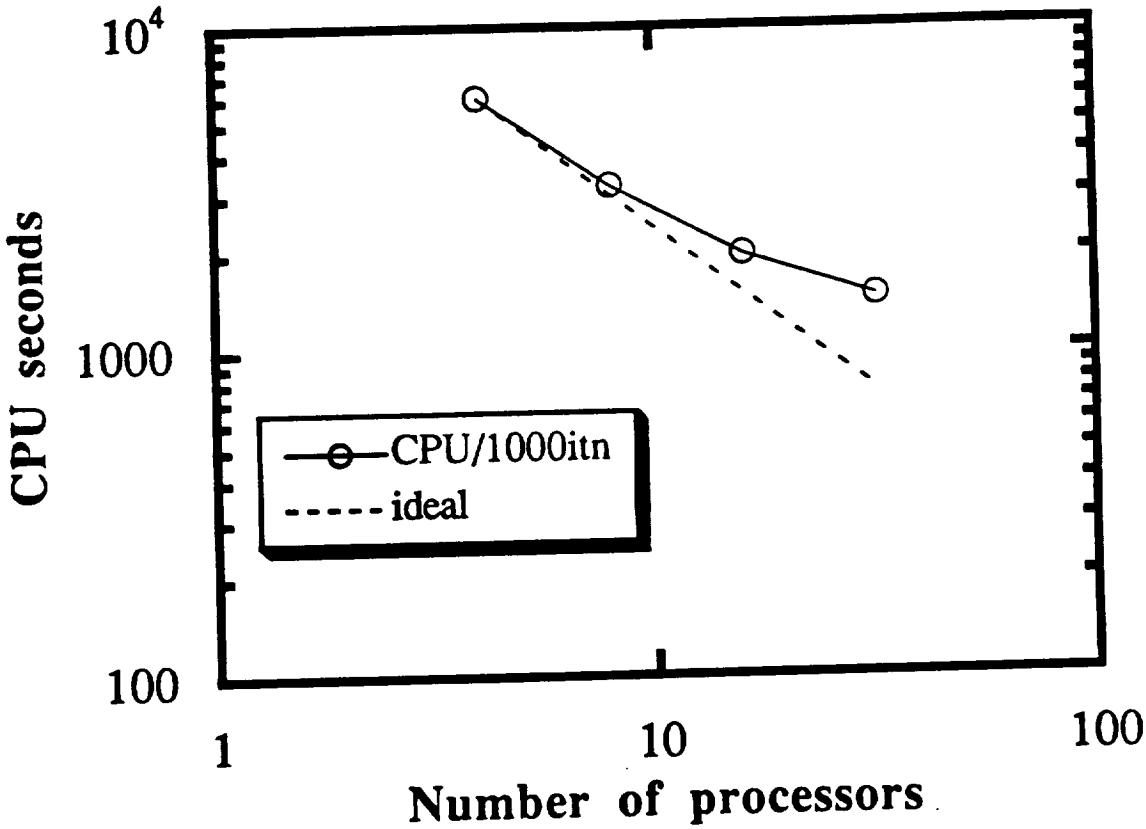

Figure 32
Comparison of the Speedup Achieved to the Ideal Speedup
(CPU Time Required for 1000 Iterations)

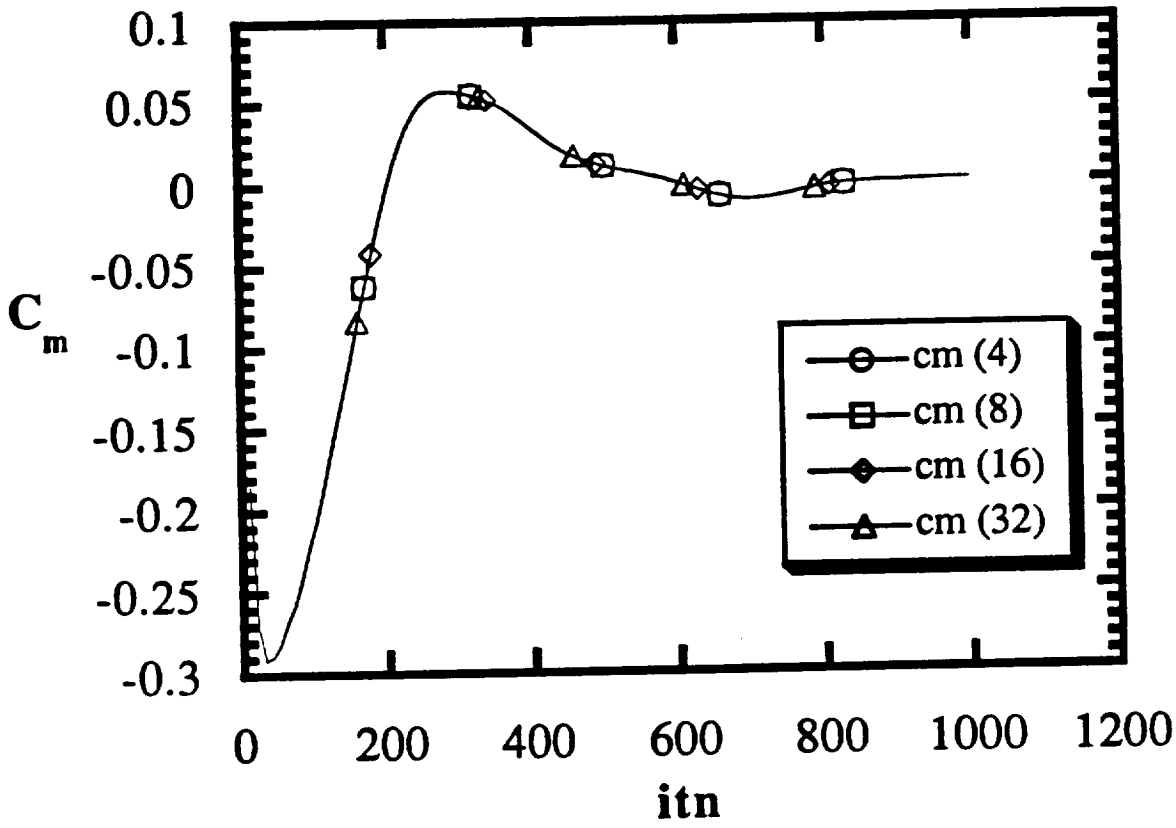

Figure 33
Comparison of the Moment Coefficient History for the Parallel Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil ($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

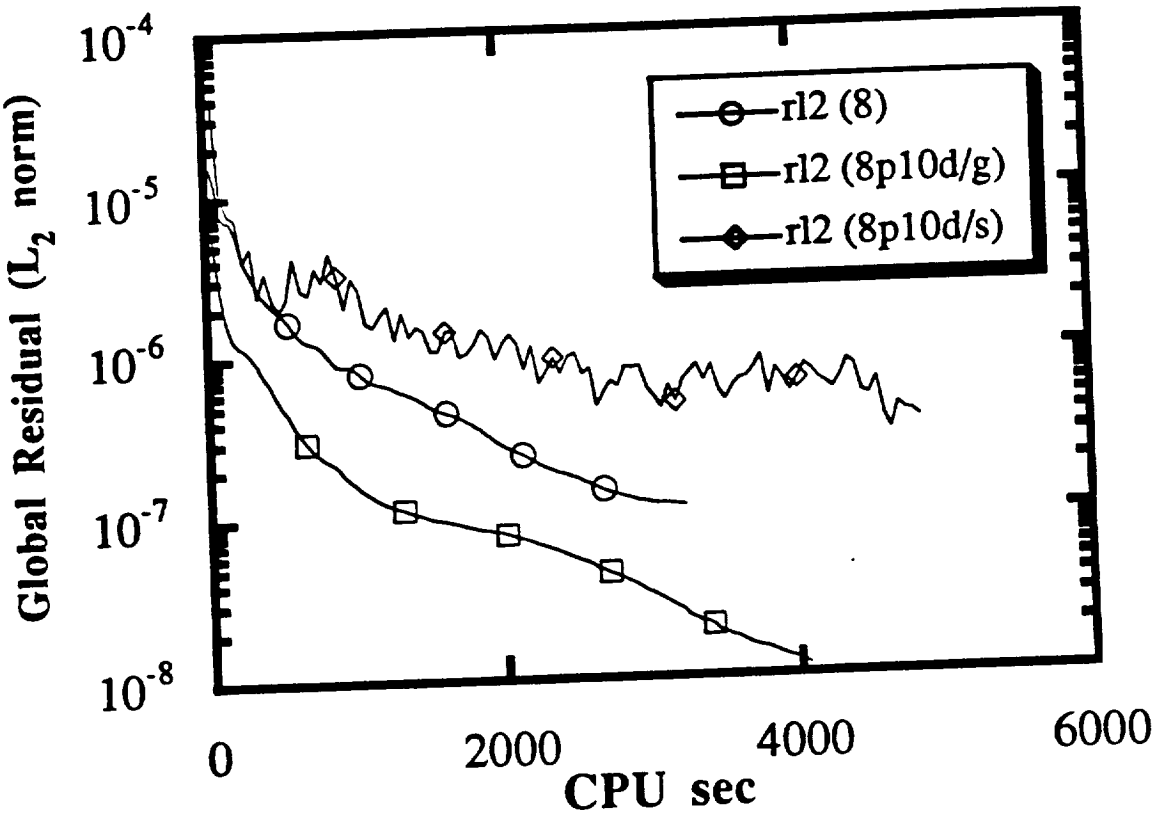

Figure 34
Comparison of the Global Residual History for the Parallel GMRES Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil ($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

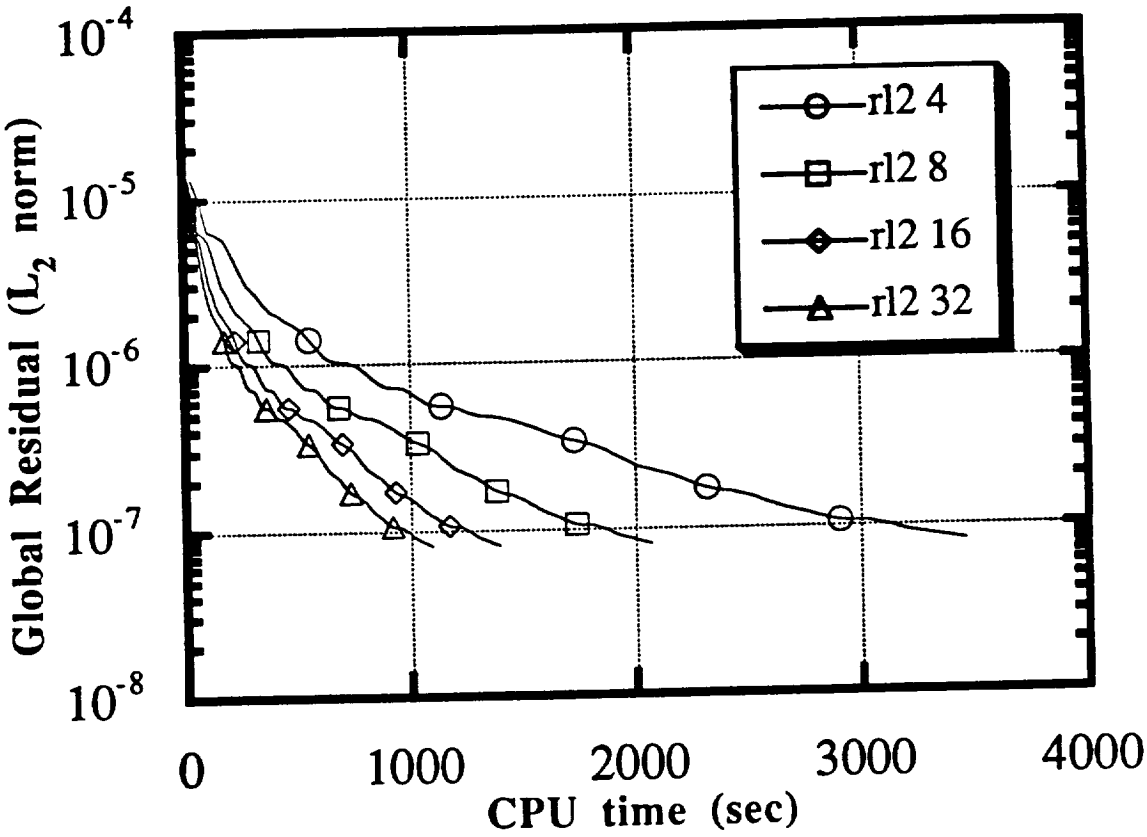

Figure 35
Comparison of the Global Residual History for the Parallel Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil Using Block Cyclic Reduction
($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

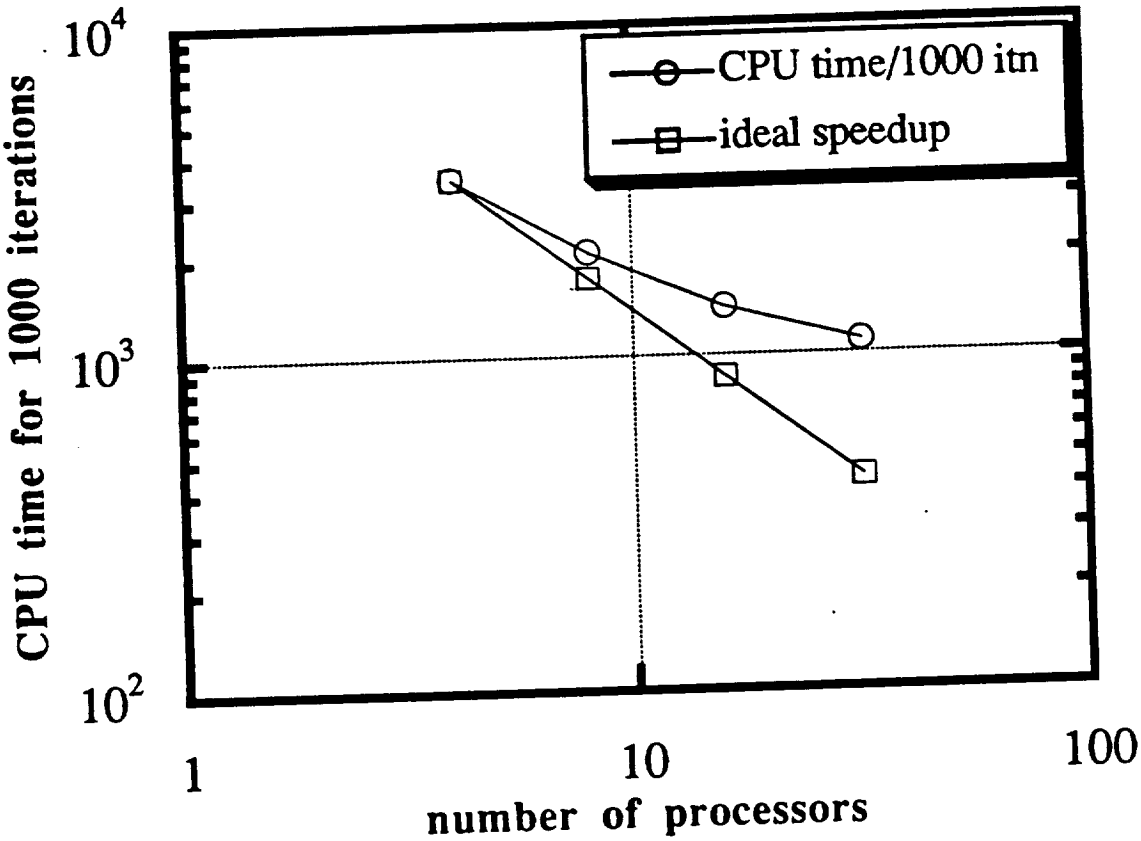

Figure 36
Comparison of the Speedup Achieved to the Ideal Speedup with Block Cyclic Reduction
(CPU Time Required for 1000 Iterations)

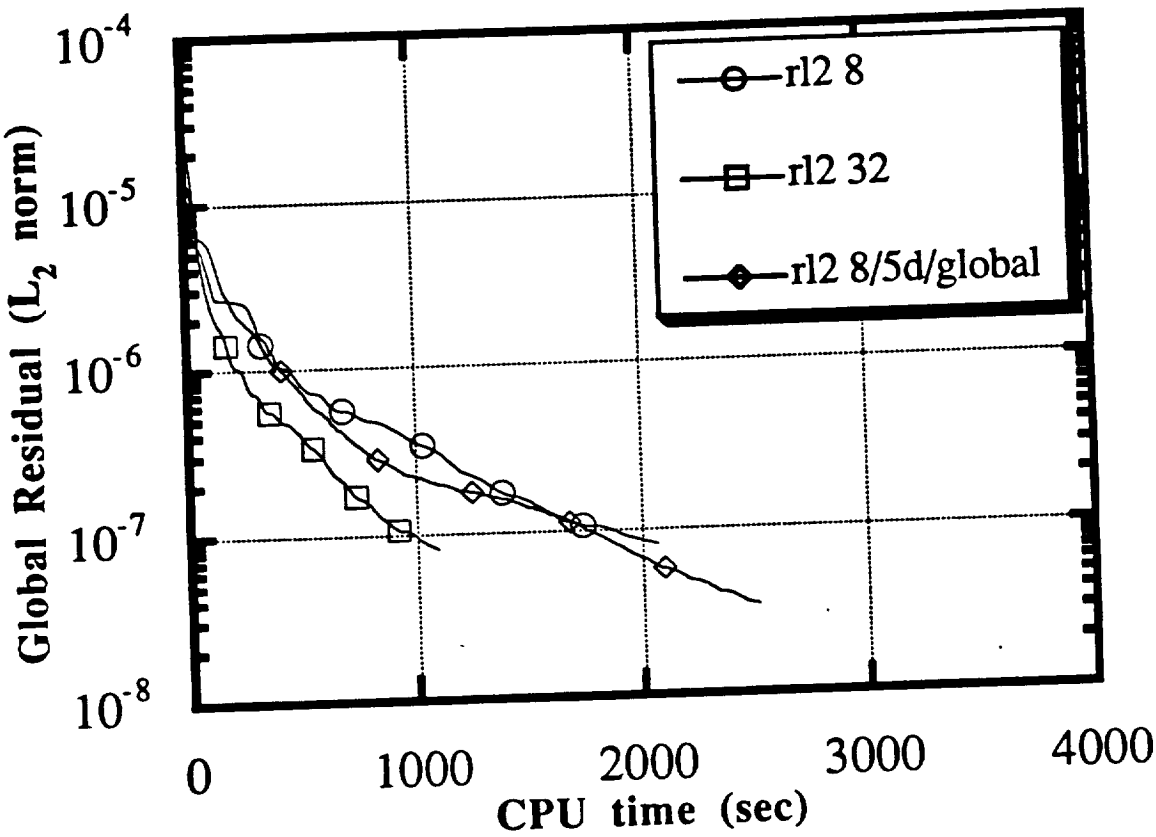

Figure 37
Comparison of the Global Residual History for the Parallel GMRES Calculation of a Steady Viscous Flow about a NACA 0012 Airfoil Using Block Cyclic Reduction
($M_\infty = 0.283$; $\alpha = 5.0°$; Re = 3,450,000)

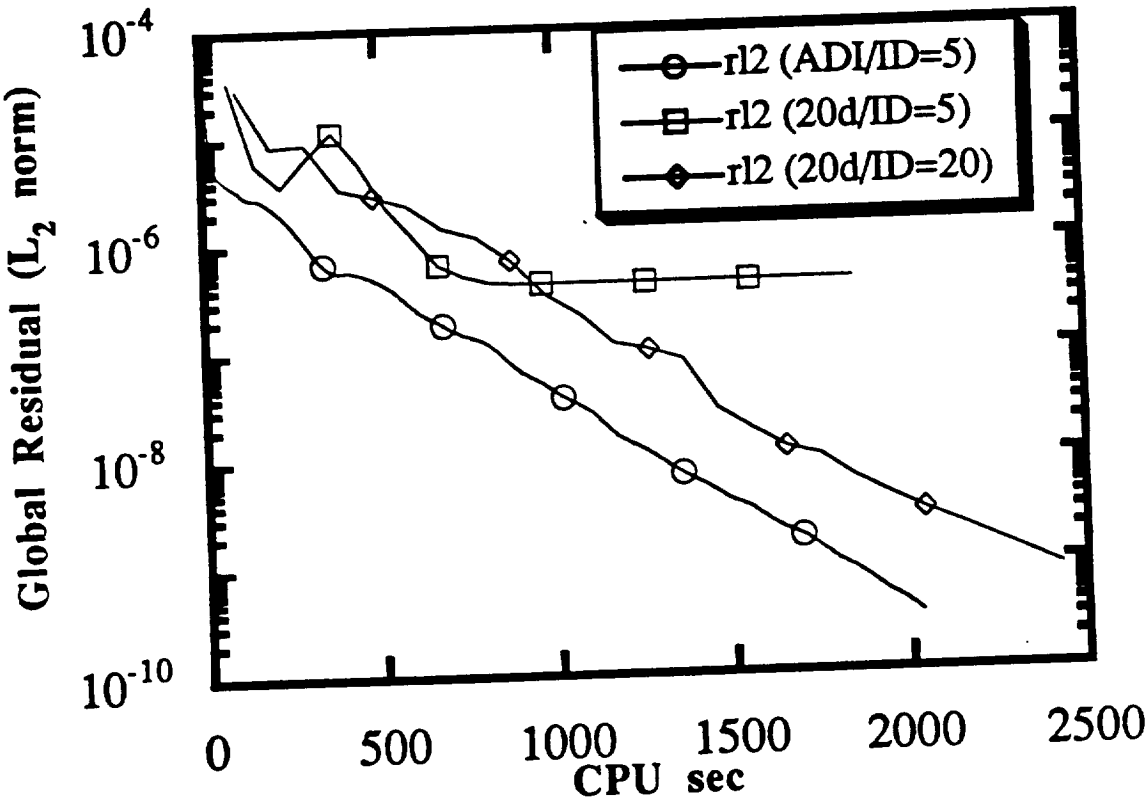

Figure 38
Effect of the Implicit Dissipation Coefficient on the GMRES Calculation of a Steady Viscous Flow about an F5 Wing ($M_\infty = 0.9$; $\alpha = 0.0°$; Re = 11,000,000)

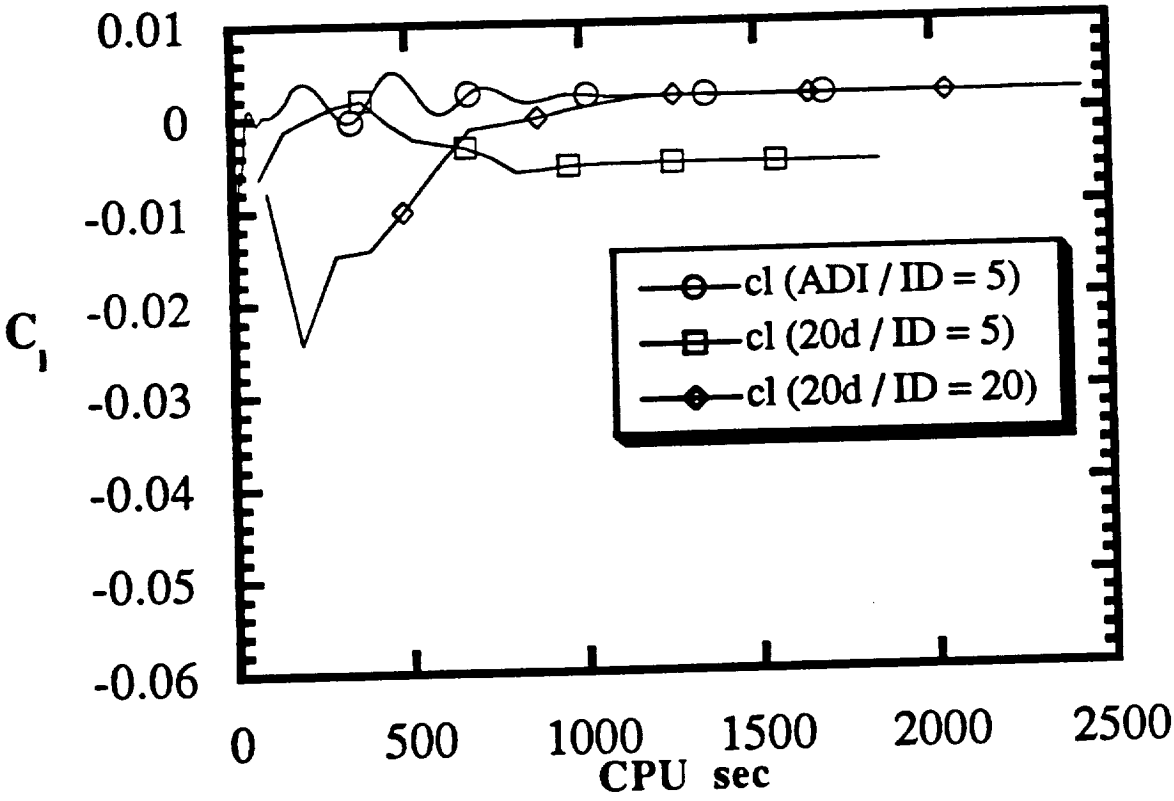

Figure 39
Comparison of the Mid-Half Span Lift Coefficient for the GMRES Calculation of a Steady Viscous Flow about an F5 Wing
($M_\infty = 0.9$; $\alpha = 0.0°$; Re = 11,000,000)

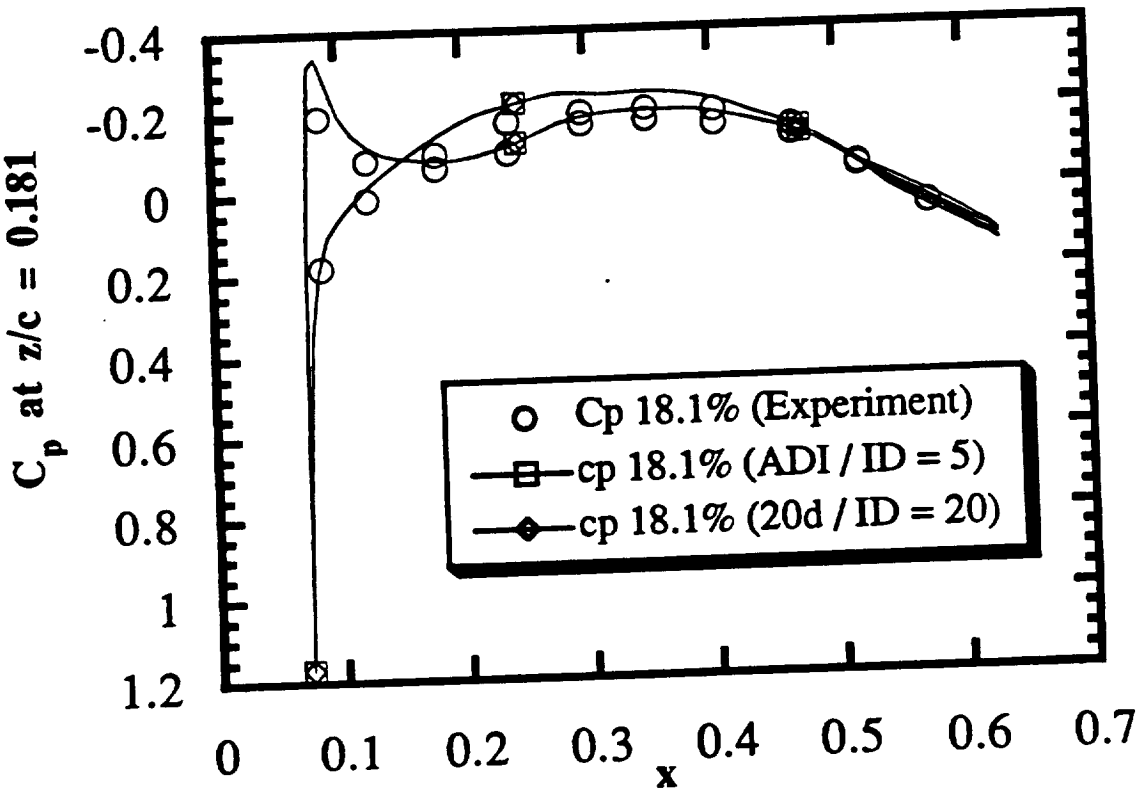

Figure 40
Comparison of the Pressure Coefficient for the GMRES Calculation of the Steady Viscous Flow about an F5 Wing
($M_\infty = 0.9$; $\alpha = 0.0°$; Re = 11,000,000; z = 0.181)

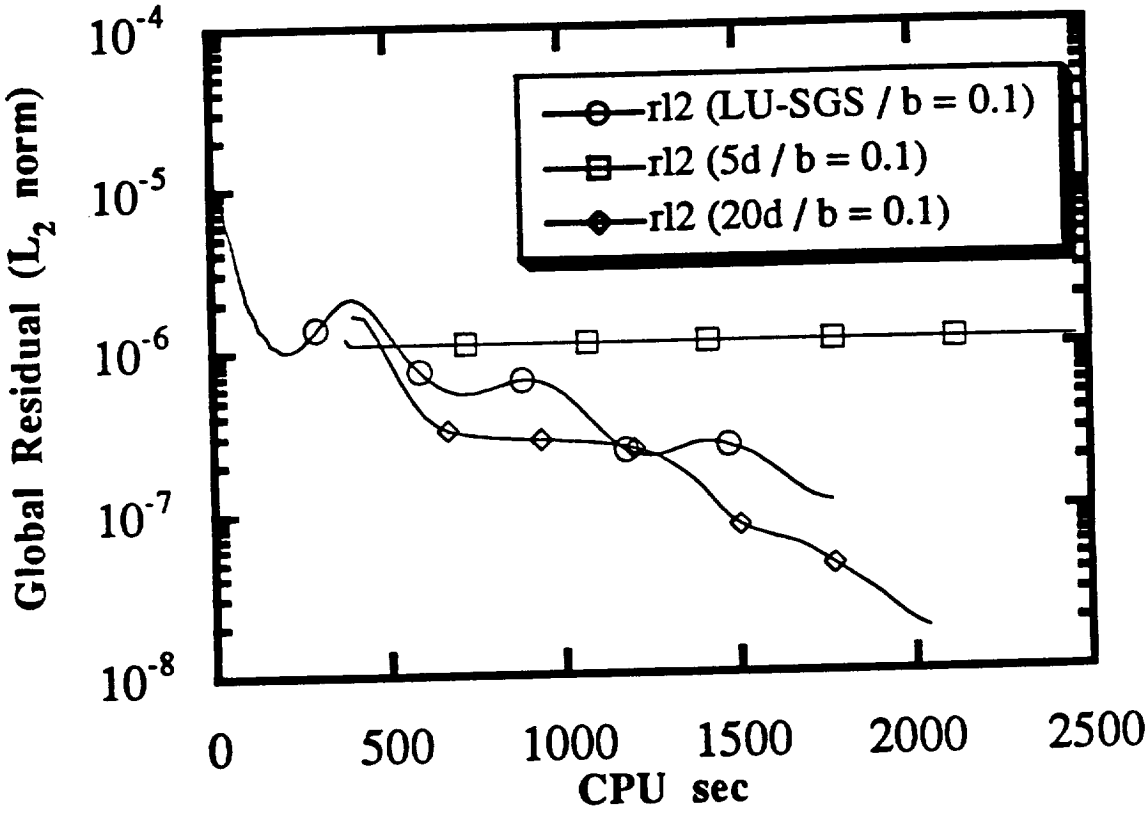

Figure 41
Comparison of GMRES / LU-SGS Global Residual Histories for an F5 Wing in Inviscid Transonic Flow
($M_\infty = 0.9$; $\alpha = 0.0°$)

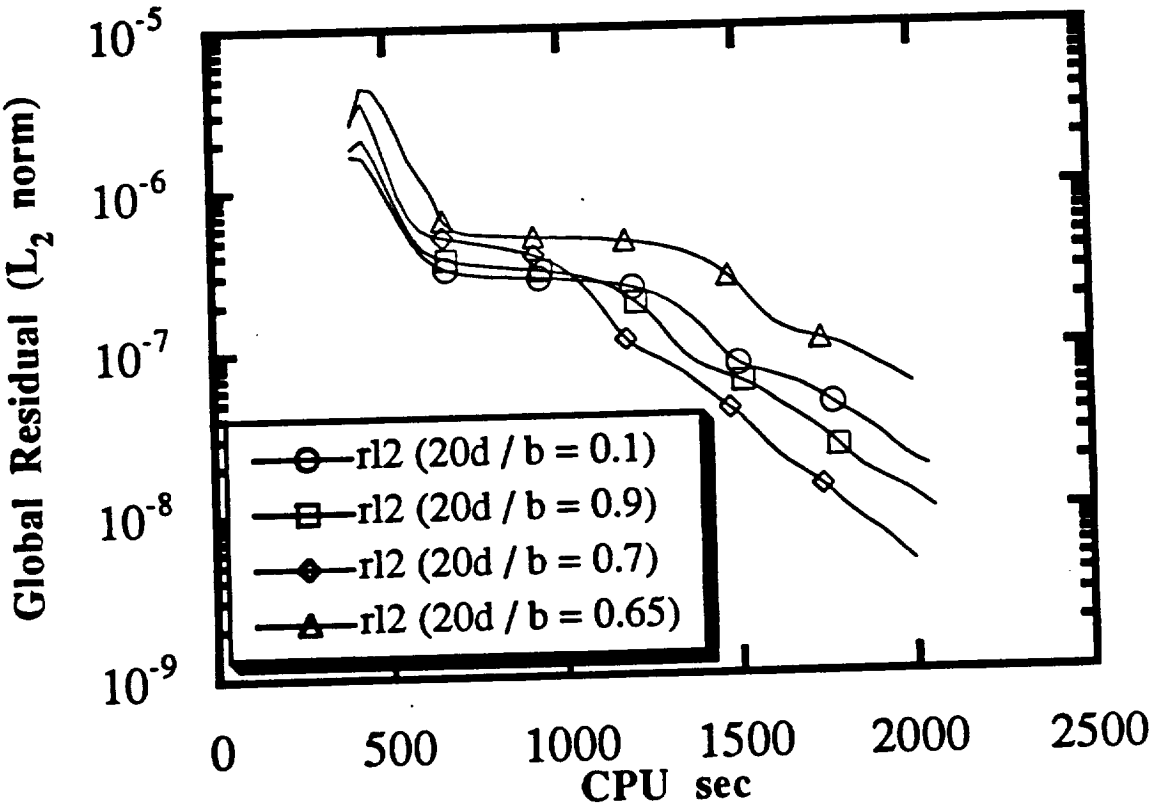

Figure 42
Effect of β Parameter on GMRES (20) Computations of Steady Inviscid Transonic Flow about an F5 Wing
($M_\infty = 0.9$; $\alpha = 0.0°$)

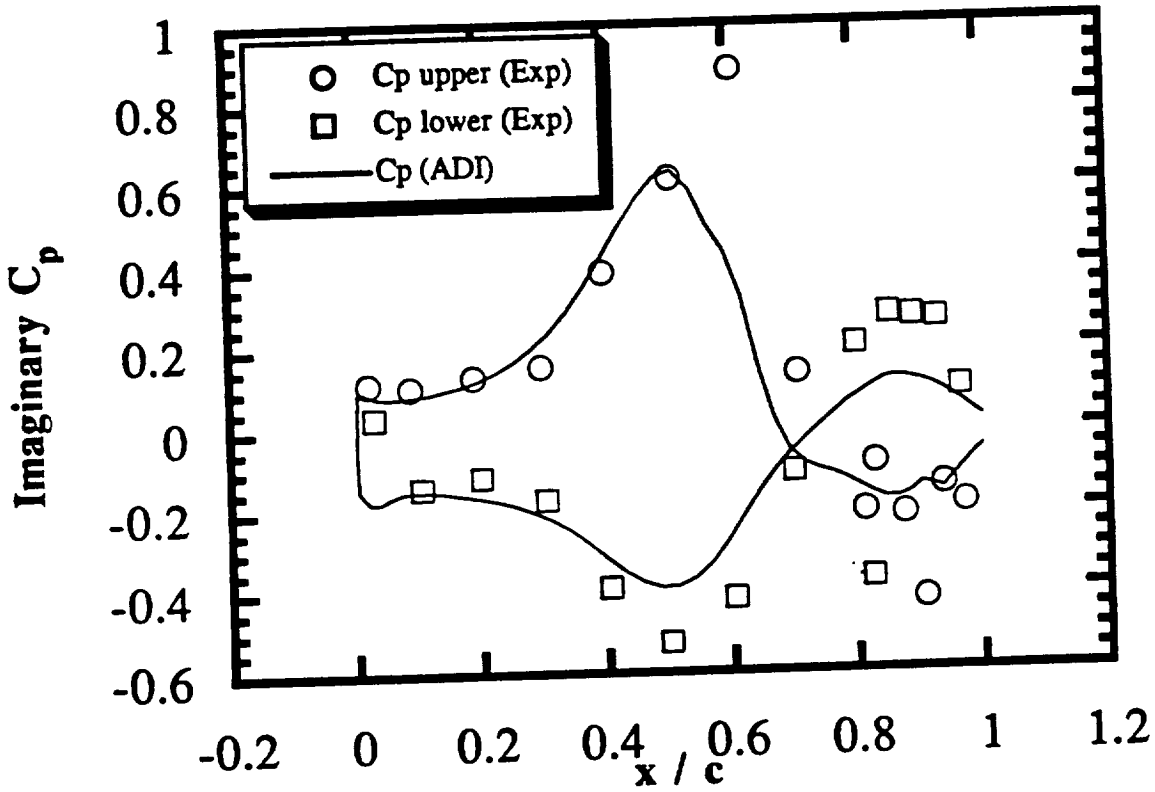

Figure 43
Comparison of 3-D ADI with Experimental Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing with an Oscillating Trailing Edge Flap
($M_\infty = 0.9$; f = 20 Hz; Re = 11,000,000; z = 0.181)
($\alpha = 0.0°$; $\alpha_{flap} = 0.5°$)

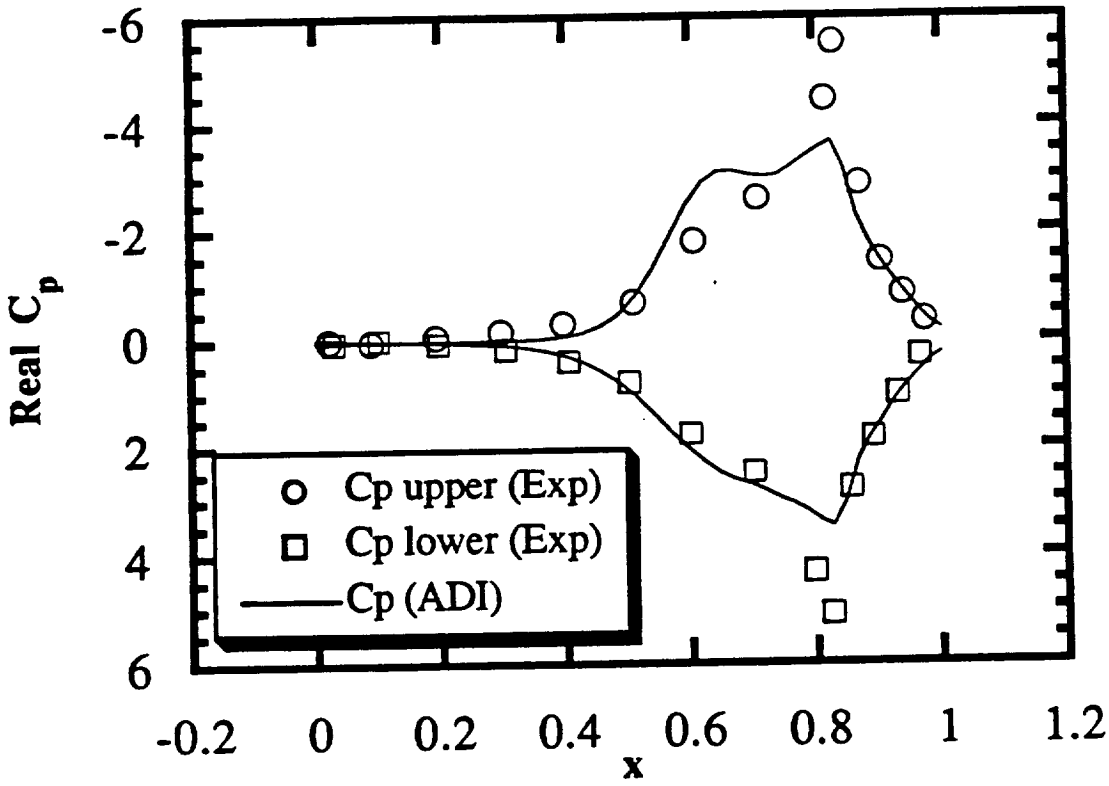

Figure 44
Comparison of 3-D ADI with Experimental Results for the Real Component of the Pressure Coefficient on an F5 Wing with an Oscillating Trailing Edge Flap
($M_\infty = 0.9$; f = 20 Hz; Re = 11,000,000; z = 0.181)
($\alpha = 0.0°$; $\alpha_{flap} = 0.5°$)

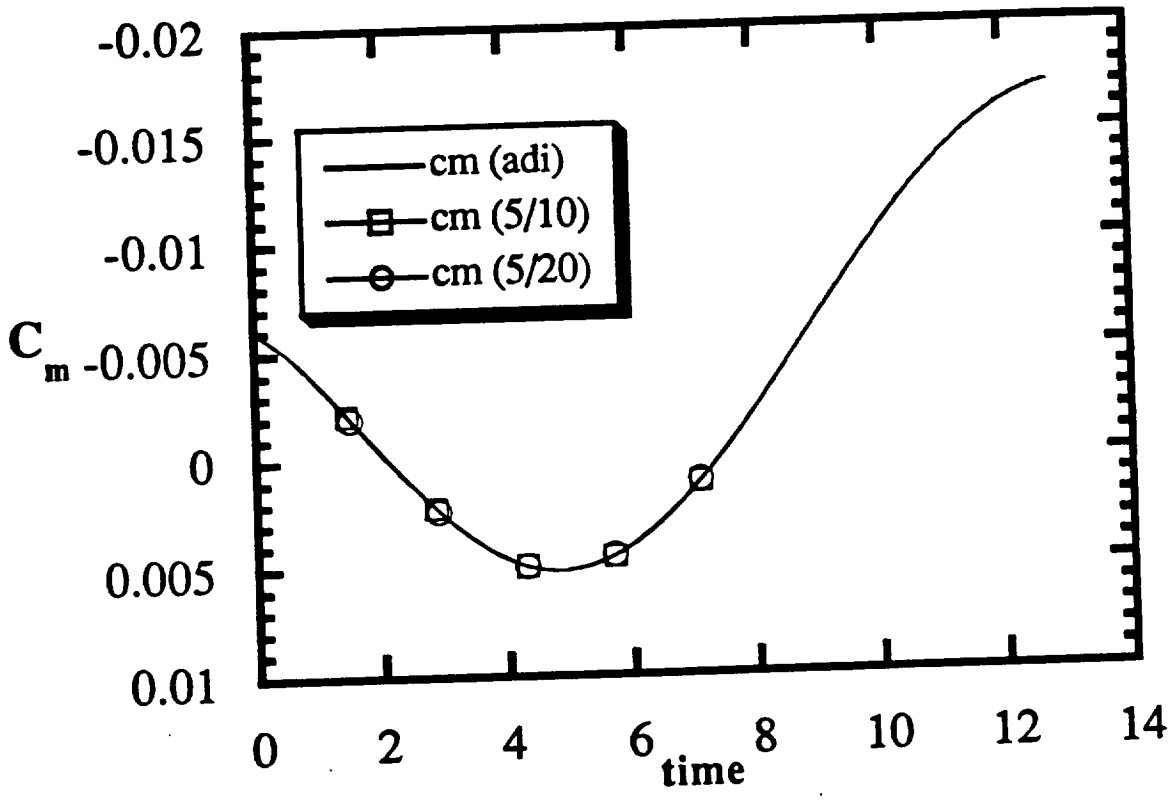

Figure 45
Comparison of 3-D GMRES with ADI Results for the Moment Coefficient on an F5 Wing with an Oscillating Trailing Edge Flap
($M_\infty = 0.9$; f = 20 Hz; Re = 11,000,000; z = 0.181)
($\alpha = 0.0°$; $\alpha_{flap} = 0.5°$)

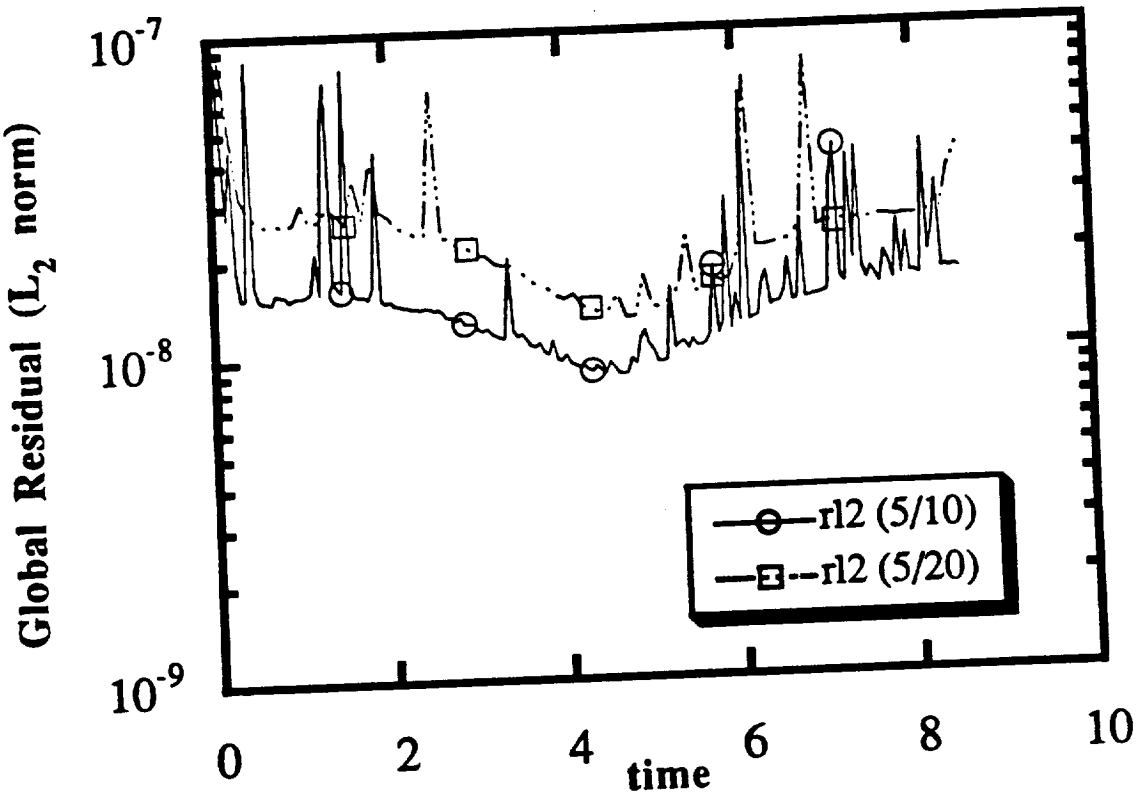

Figure 46
Effect of Time Step on the GMRES (5/x) Results for the Global Residual of the Transonic Viscous Flow about an F5 Wing with an Oscillating Trailing Edge Flap
($M_\infty = 0.9$; f = 20 Hz; Re = 11,000,000; z = 0.181)
($\alpha = 0.0°$; $\alpha_{flap} = 0.5°$)

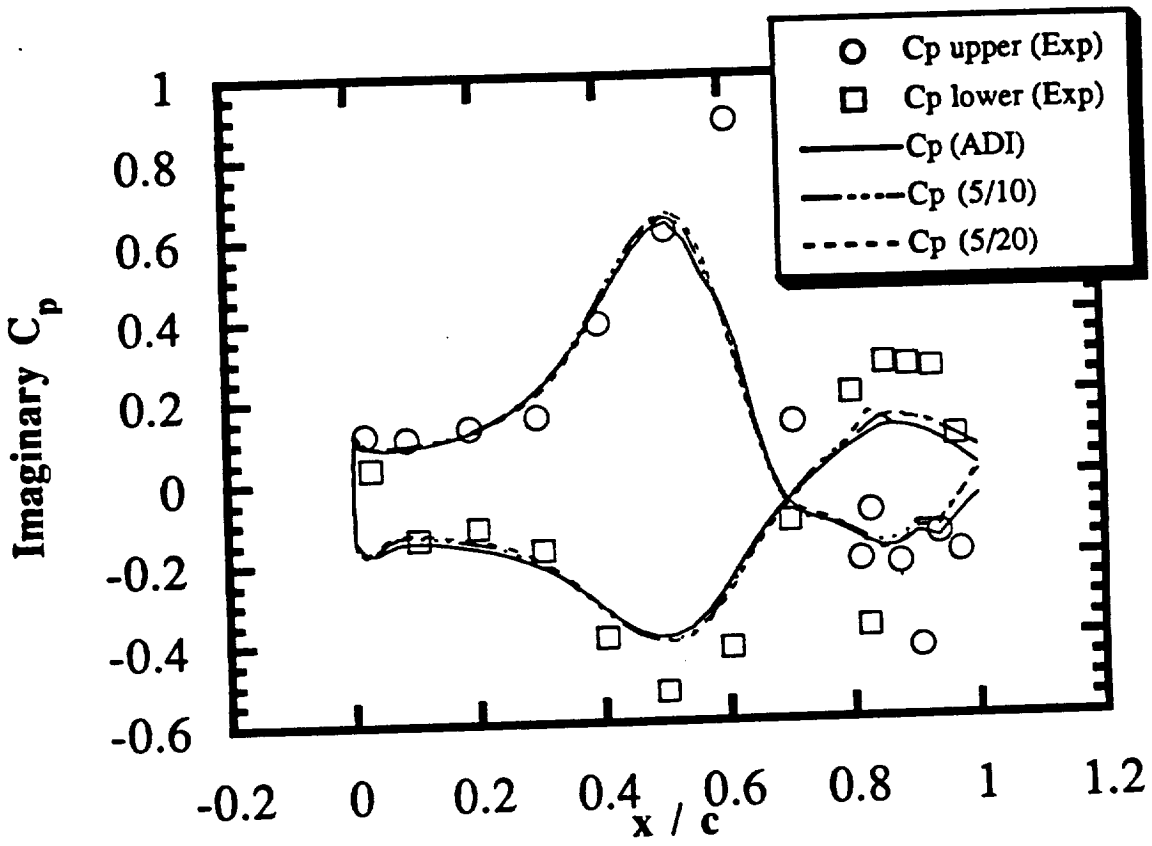

Figure 47
Comparison of 3-D GMRES Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing with an Oscillating Trailing Edge Flap
($M_\infty = 0.9$; f = 20 Hz; Re = 11,000,000; z = 0.181)
($\alpha = 0.0°$; $\alpha_{flap} = 0.5°$)

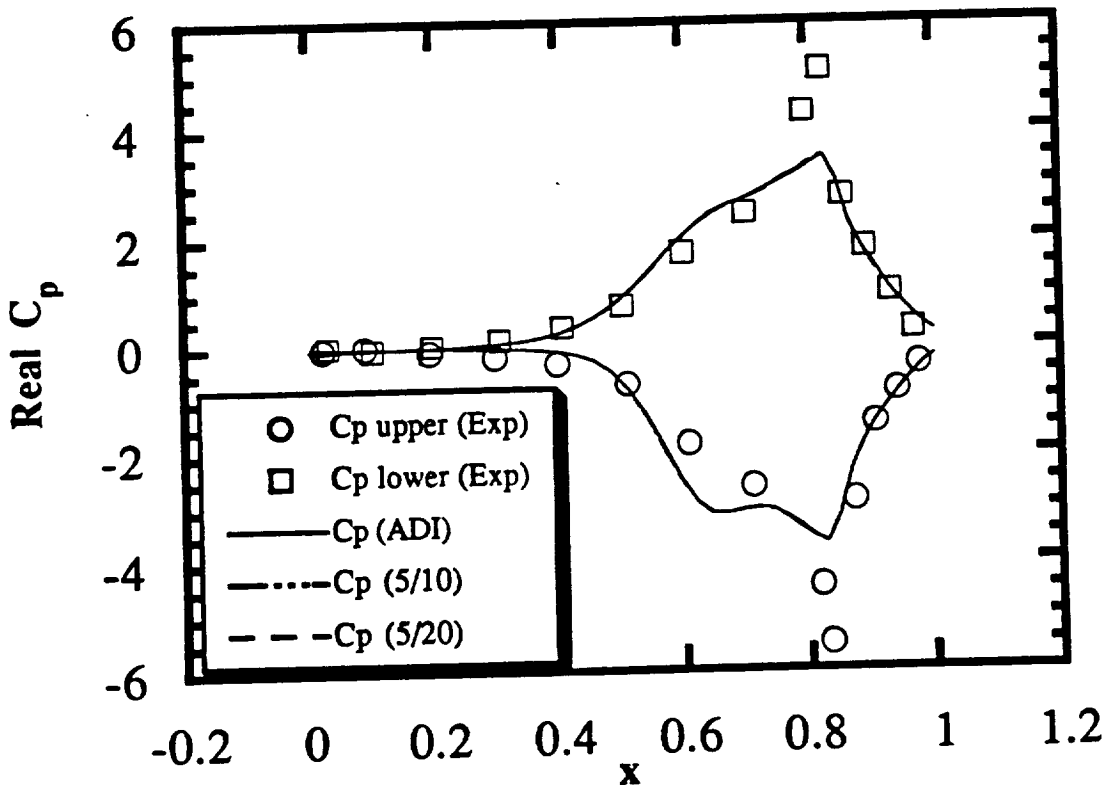

Figure 48
Comparison of 3-D GMRES Results for the Real Component of the Pressure Coefficient on an F5 Wing with an Oscillating Trailing Edge Flap
($M_\infty = 0.9$; f = 20 Hz; Re = 11,000,000; z = 0.181)
($\alpha = 0.0°$; $\alpha_{flap} = 0.5°$)

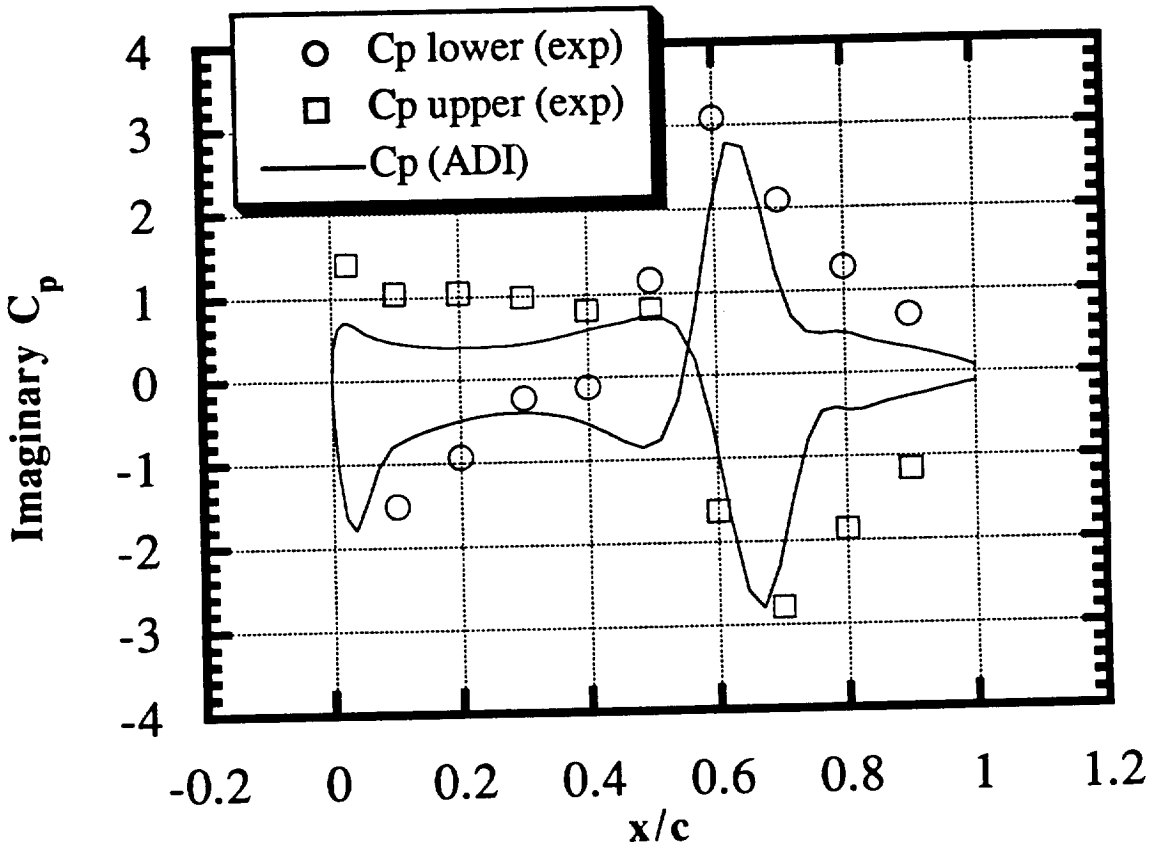

Figure 49
Comparison of 3-D ADI with Experimental Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz; z = 0.181)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$)

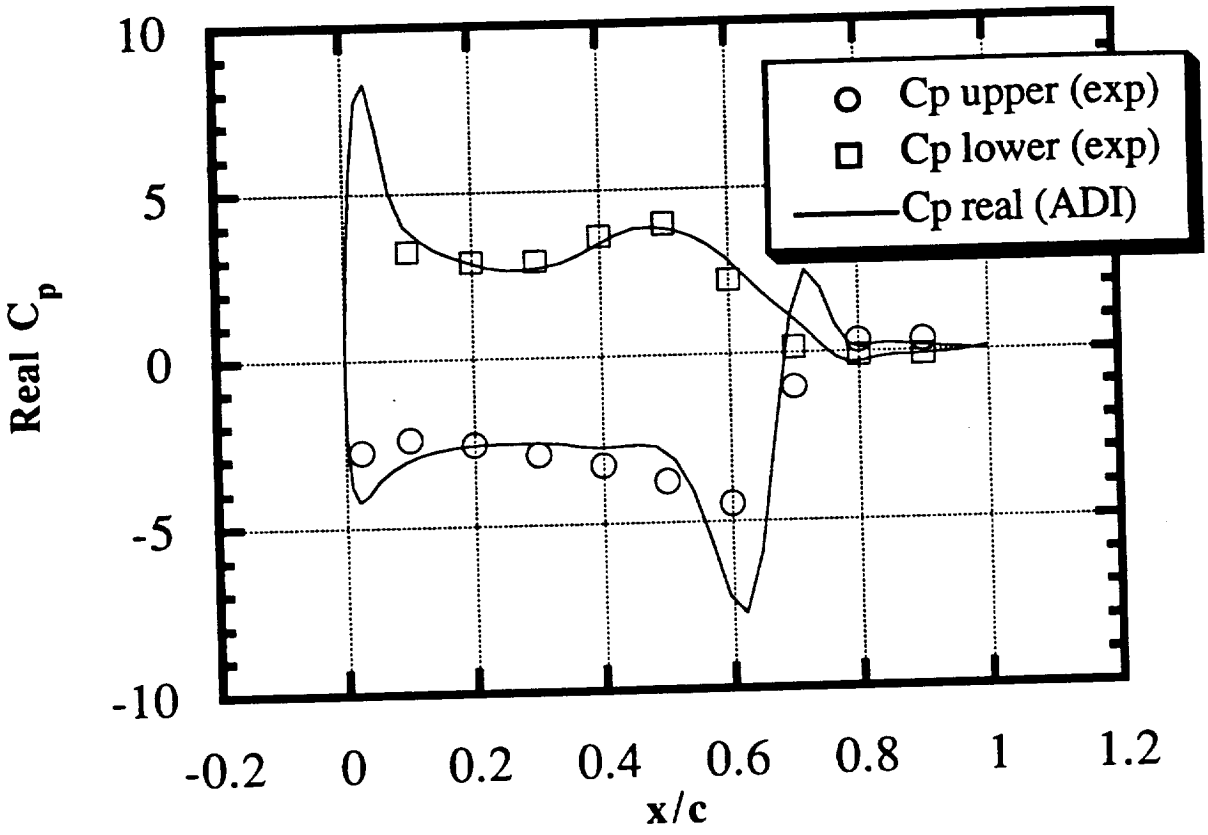

Figure 50
Comparison of 3-D ADI with Experimental Results for the Real Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz; z = 0.181)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$)

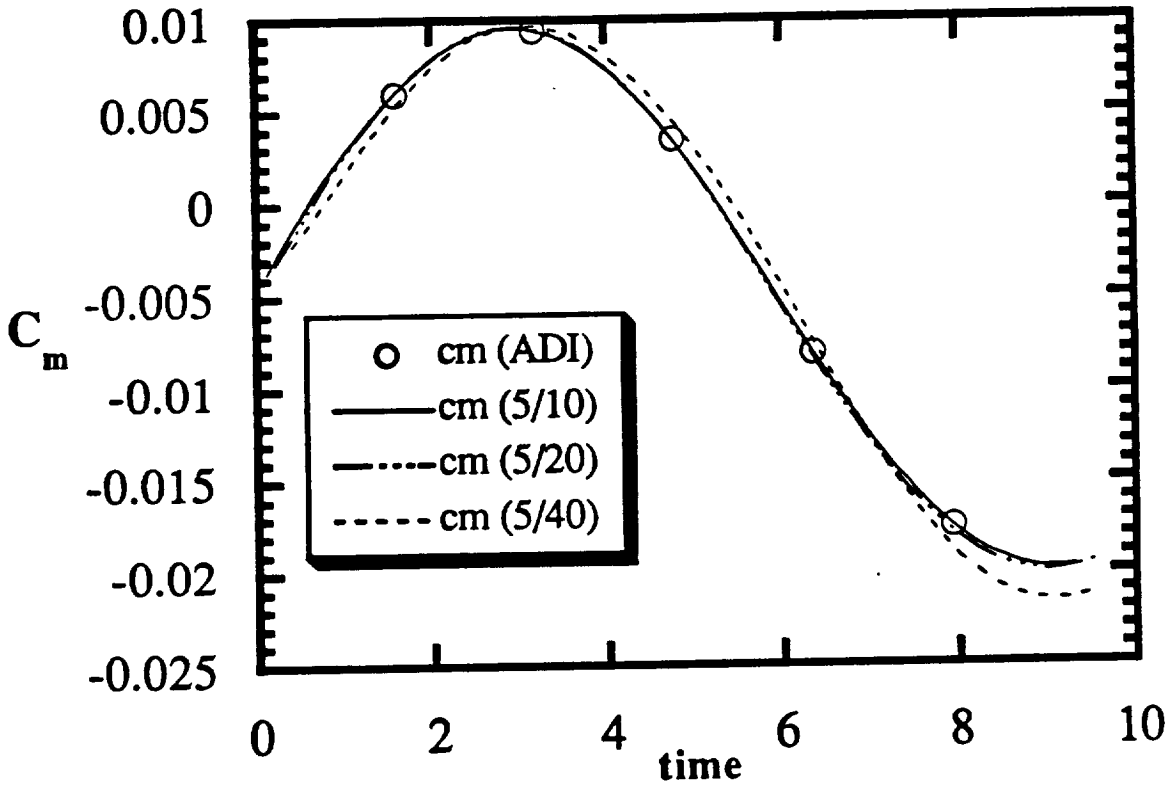

Figure 51
Comparison of 3-D GMRES (5/x) with ADI Results for the Mid-Half Span Moment Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$)

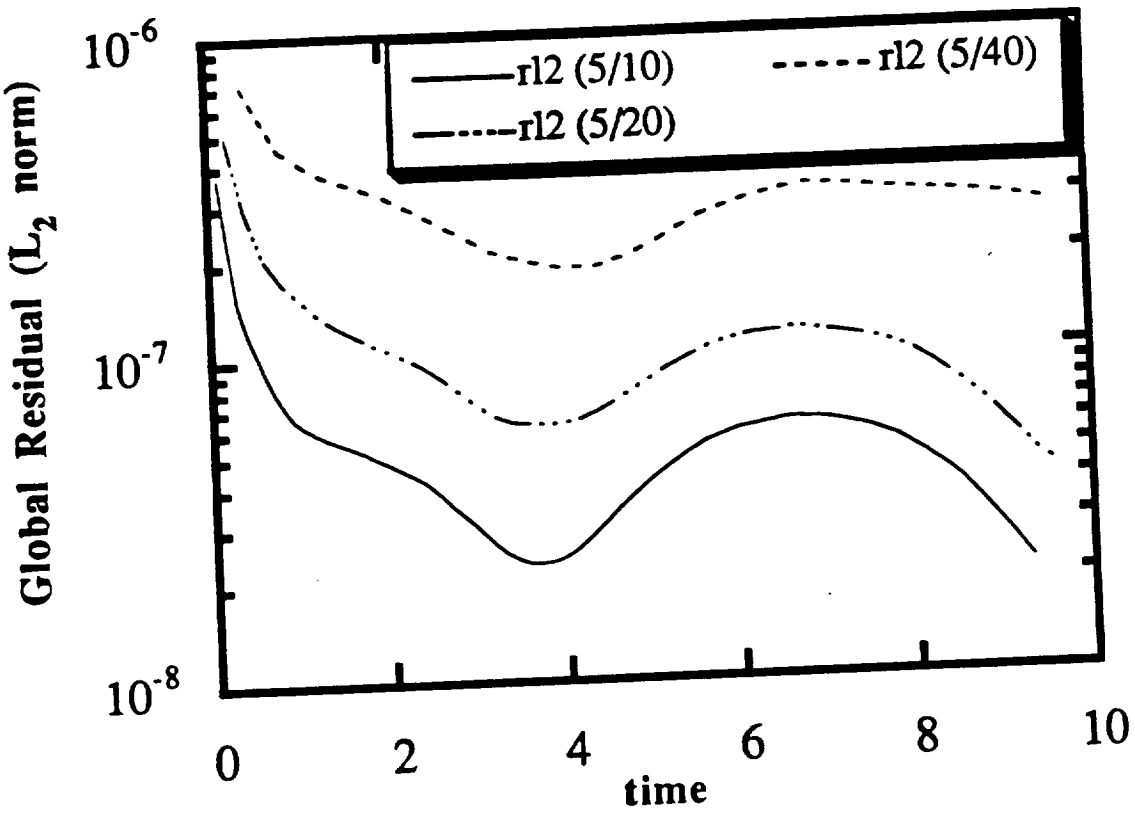

Figure 52
Comparison of 3-D GMRES (5/x) Global Residual Histories for the Inviscid Flow about an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$)

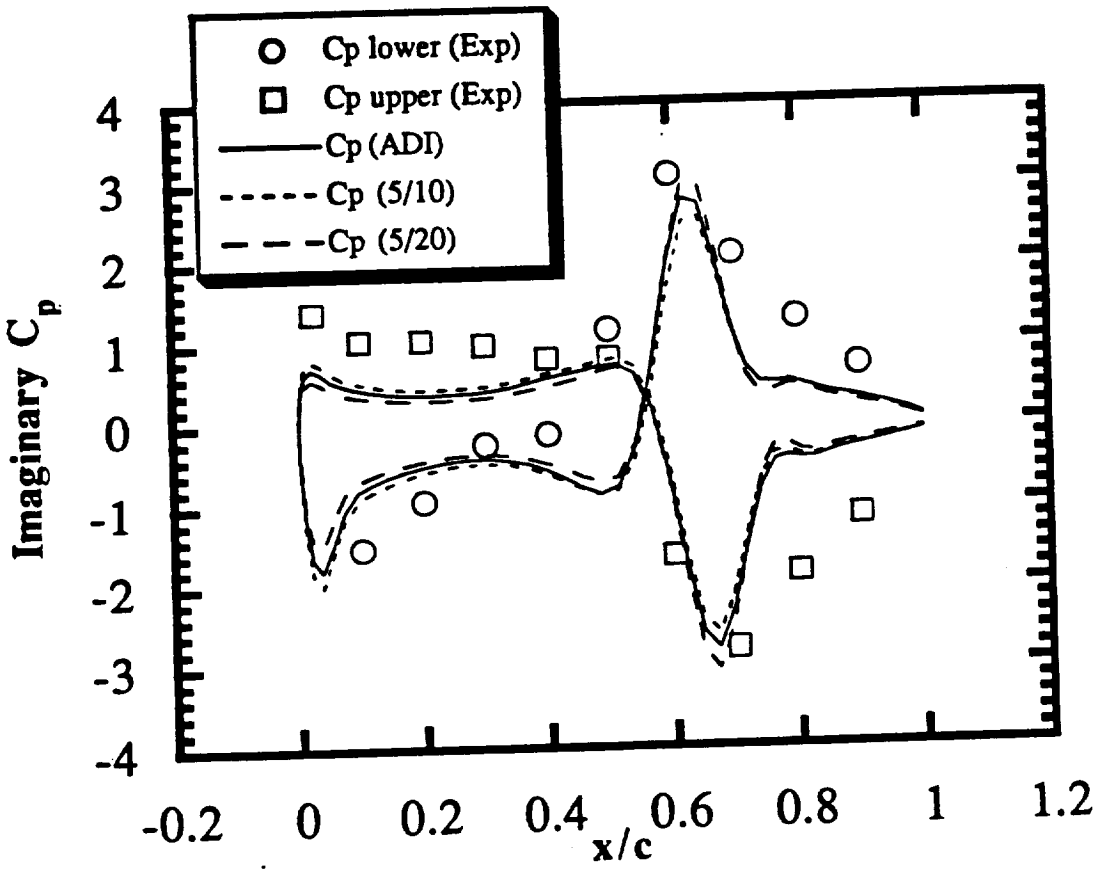

Figure 53
Comparison of 3-D GMRES (5/x) Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

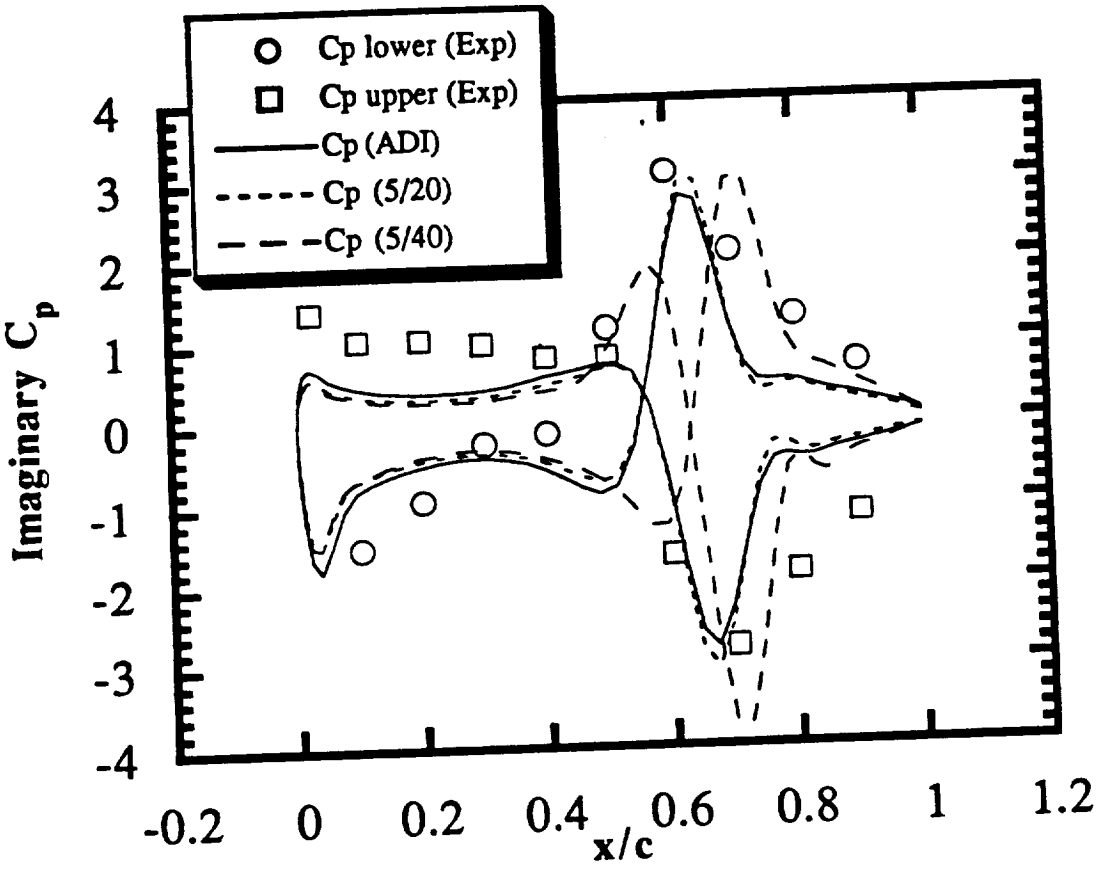

Figure 54
Comparison of 3-D GMRES (5/x) Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

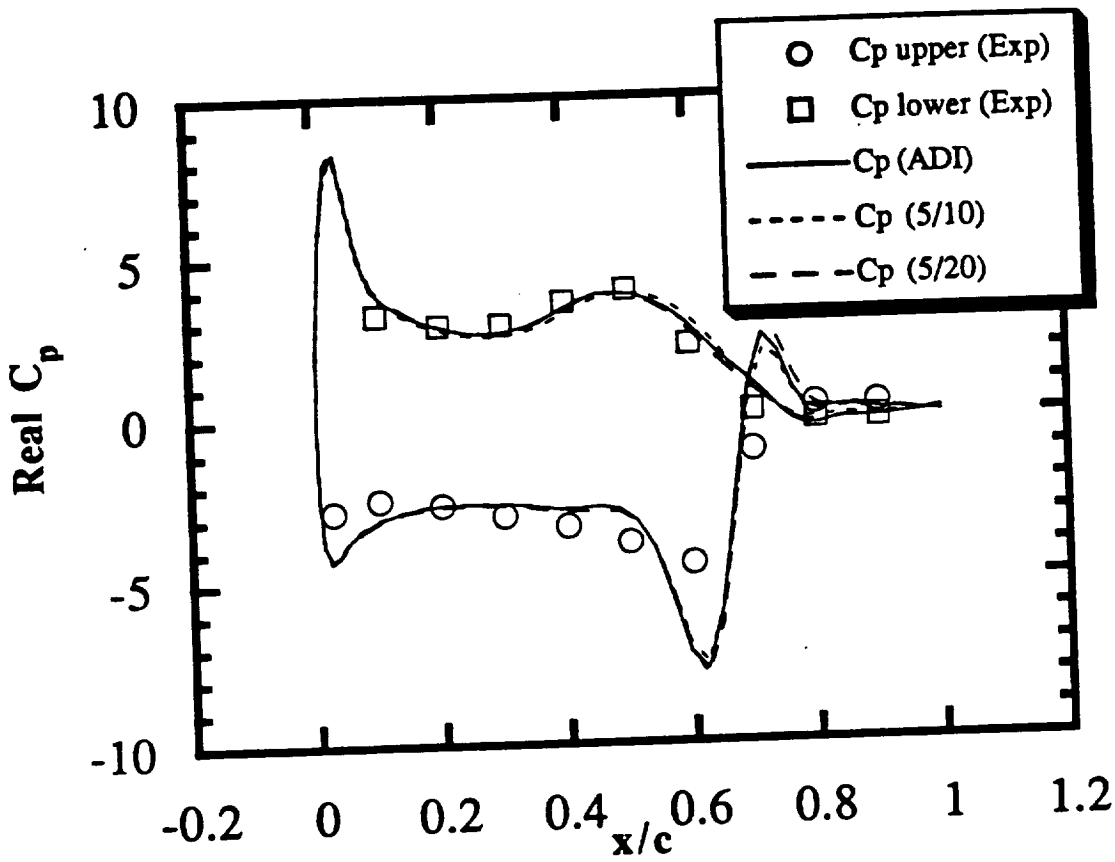

Figure 55
Comparison of 3-D GMRES (5/x) Results for the Real Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

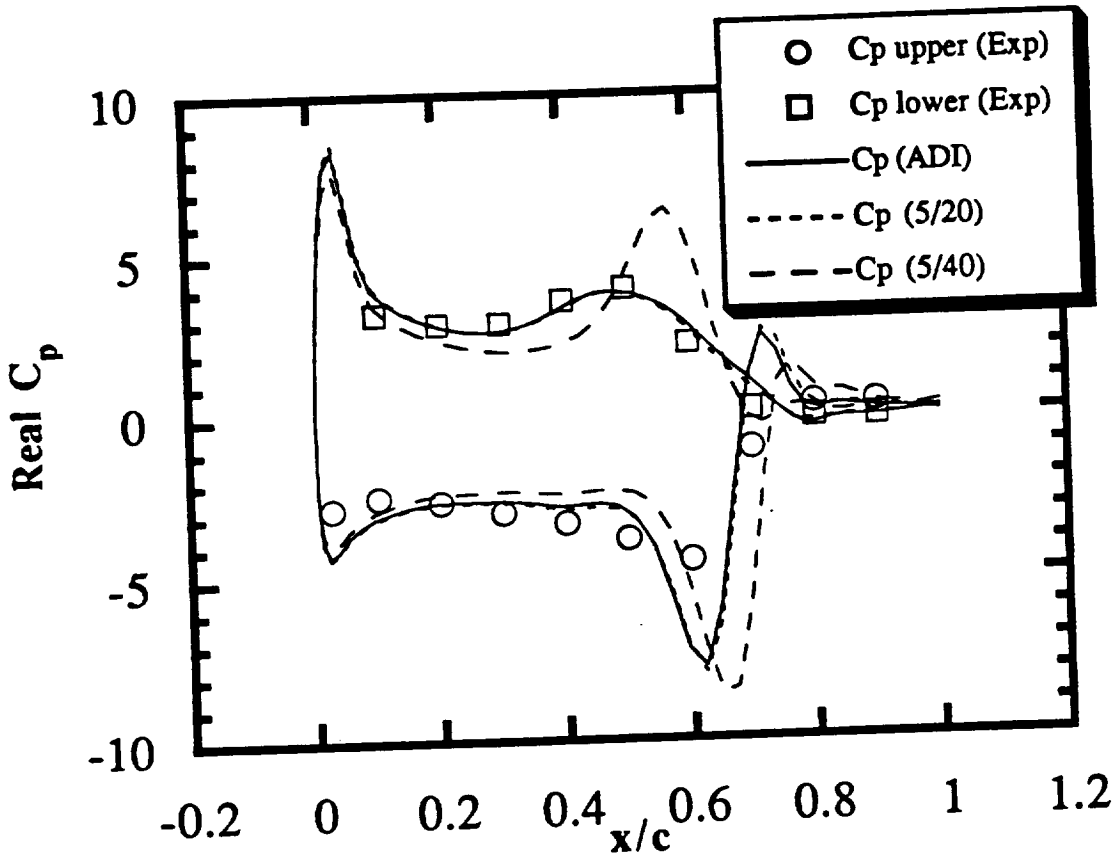

Figure 56
Comparison of 3-D GMRES (5/x) Results for the Real Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

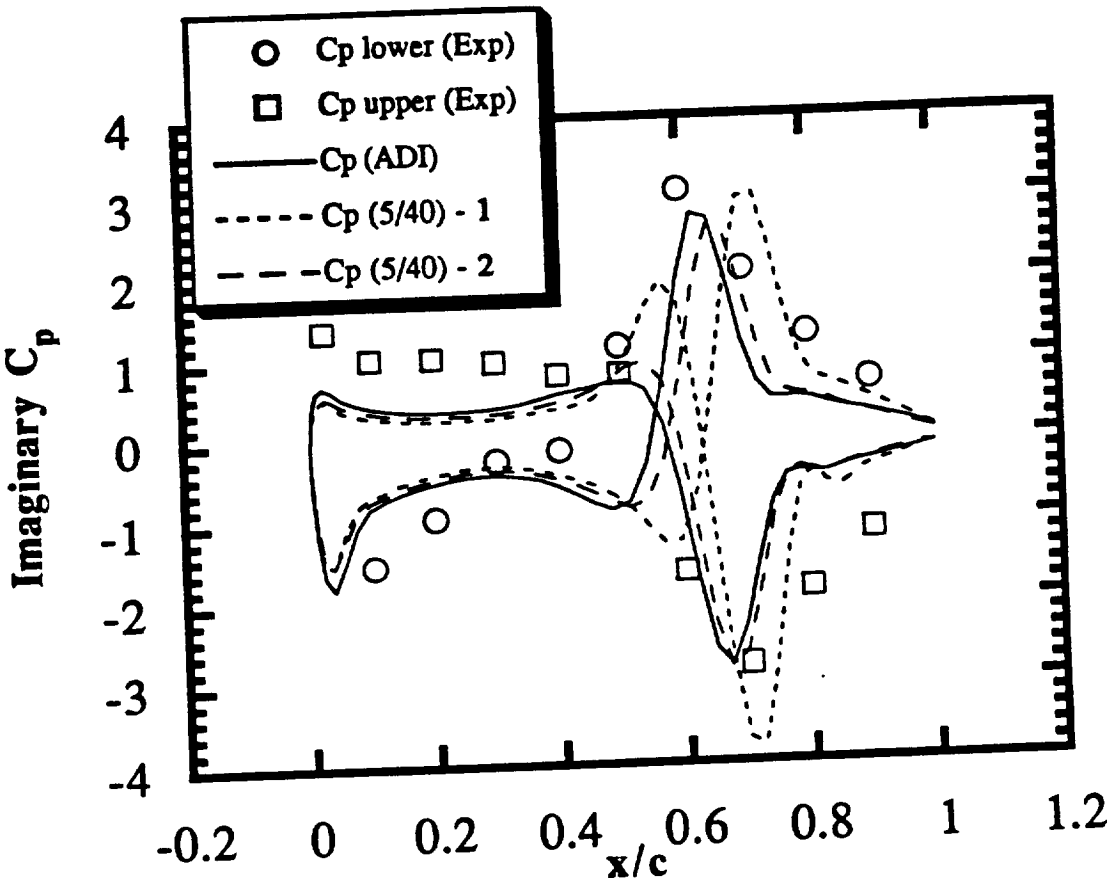

Figure 57
Comparison of 3-D GMRES (5/40) Restart Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

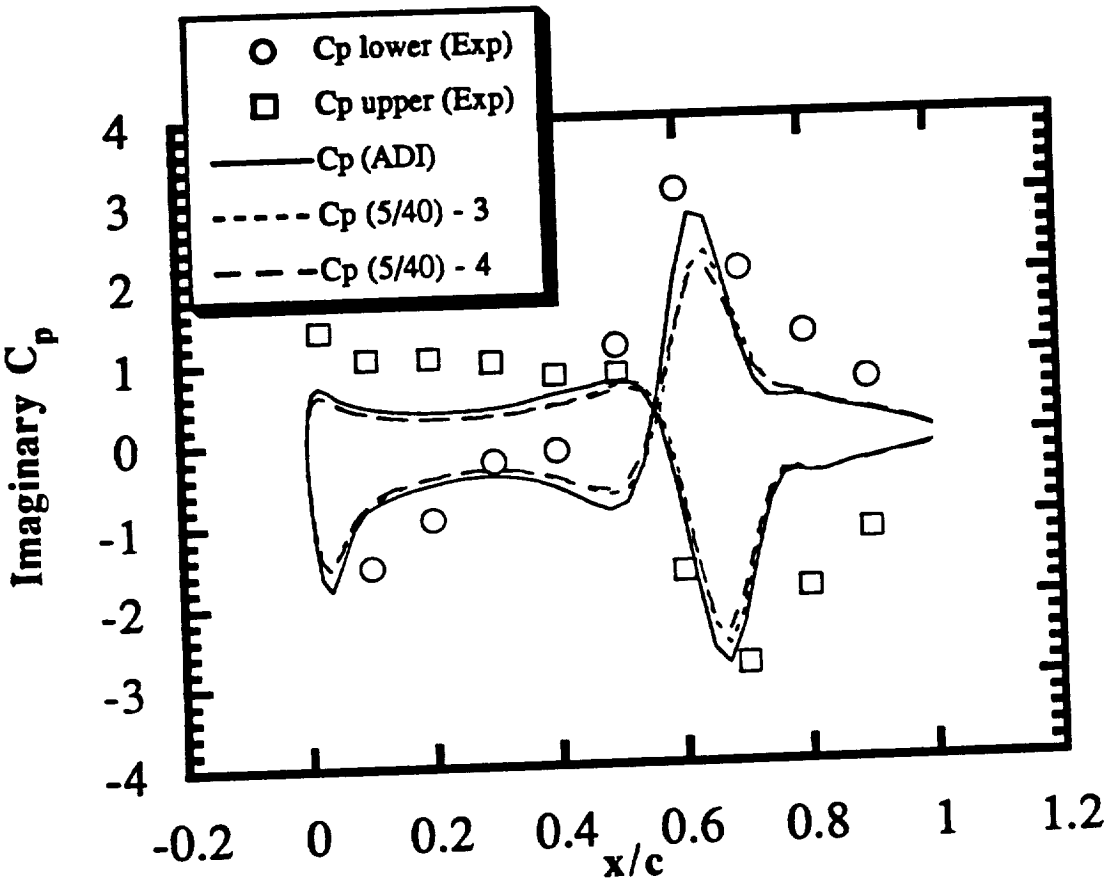

Figure 58
Comparison of 3-D GMRES (5/40) Restart Results for the Imaginary Component of the Pressure Coefficient on an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

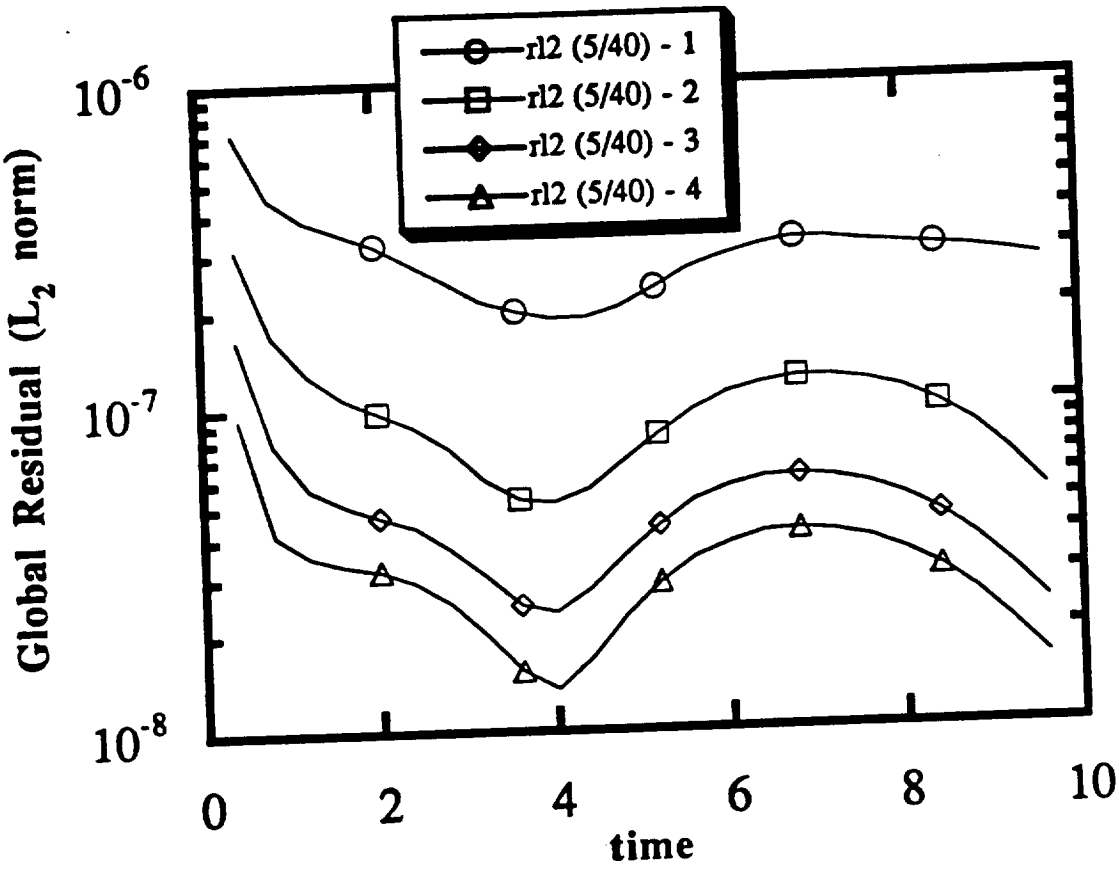

Figure 59
Comparison of 3-D GMRES (5/40) Restart Results for the Global Residual of the Inviscid Flow about an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)

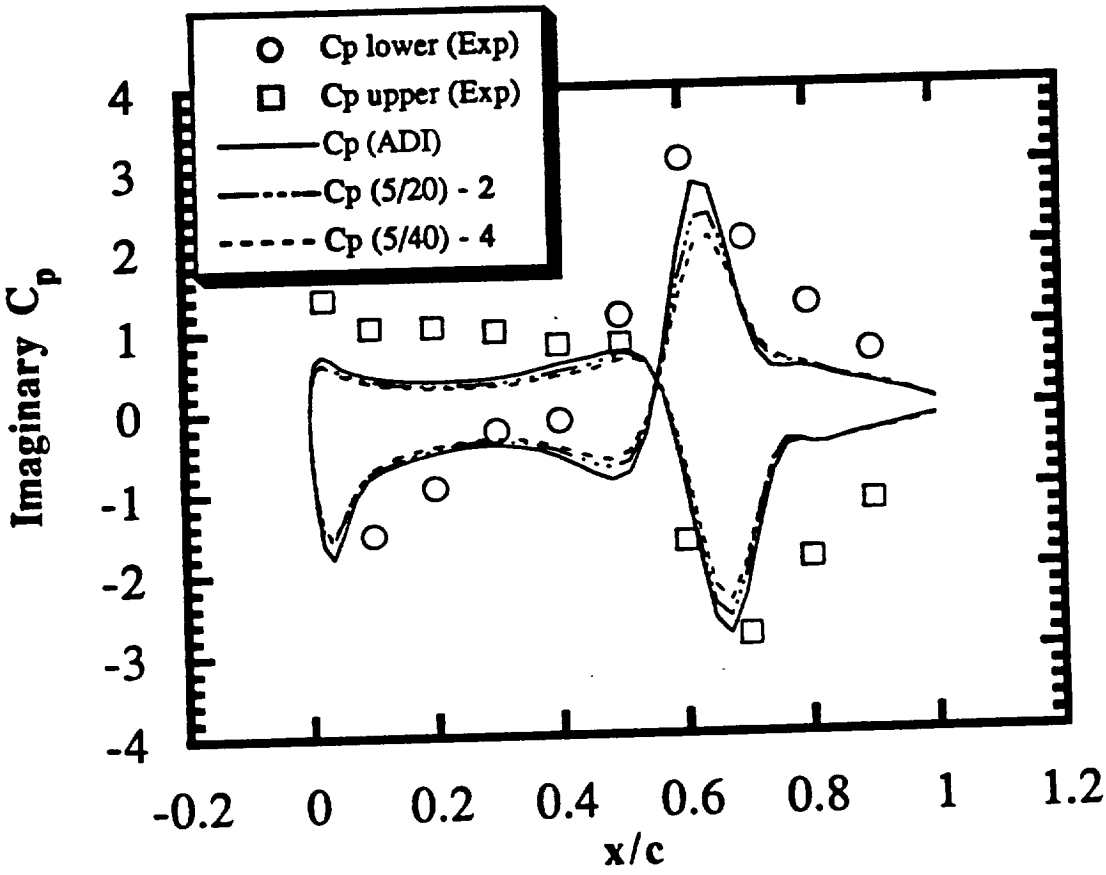

Figure 60
Effect of Time Step on the GMRES Result for the Imaginary Component of the Pressure Coefficient about an F5 Wing Undergoing Modal Vibration
($M_\infty = 0.9$; f = 40 Hz)
($\alpha = 0.0°$; $\alpha_{max} = 0.5°$; z = 0.181)